НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:

Завідувач кафедри

_____*Сергій СТІРЕНКО*

(підпис)

“ ” 20

Дипломний проект

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютерних систем»

спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Система виявлення вторгнень у мережі»

Виконав:
студент _IV_курсу, групи__ІП-62__

___Сміщенко Богдан Олександрович_____ _____
(прізвище, ім’я, по батькові) (підпис)

Керівник Доцент, к.т.н. Волокита Артем Миколайович _ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Консультант __н. контроль_ _ д.т.н., проф Симоненко В.П._ _____
(назва розділу) (вчені ступінь та звання, прізвище, ініціали) (підпис)

Рецензент доцент кафедри АУТС, к.т.н.,доц. Катін Павло Юрійович _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМ. ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____*Сергій СТІРЕНКО*
(підпис)
“ ” 20 р.

ЗАВДАННЯ

на дипломний проект студенту

Сміщенку Богдану Олександровичу

1. Тема проекту «*Система виявлення вторгнень у мережі»*

керівник проекту Волокита Артем Миколайович, доцент,

2. Термін здачі студентом закінченої роботи_____*2020р.*
3. Вихідні дані до проекту *технічне завдання, теоретичні дані.*
4. Зміст пояснювальної записки: *Огляд існуючих рішень, огляд можливих засобів створення додатків з машинним навчанням, програмна реалізація продукту*
5. Перелік графічного матеріалу (з точним позначенням обов’язкових креслень). структурна схема системи, узагальнена схема роботи системи, блок-схема алгоритму системи.

6. Консультант роботи, з вказівкою розділів роботи, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів дипломного проекту (роботи)	Строк виконання етапів проекту(роботи)	Примітки
1.	*Затвердження теми роботи*	10.12.2019 – 17.12.2019	
2.	*Вивчення та аналіз завдання*	15.12.2019-15.03.2020	
3.	*Розробка архітектури та загальної структури систем*	15.03.2020-15.04.2020	
4.	*Розробка структур окремих Підсистем*	15.04.2020 – 1.05.2020	
5.	*Програмна реалізація системи*	1.05.2020 – 14.05.2020	
6.	*Оформлення пояснювальної записки*	14.05.2020 – 25.05.2020	
7.	Захист програмного продукту		
8.	*Передзахист*		
9.	*Захист*		

Студент Богдан Сміщенко _____

(підпис)

Керівник Артем Волокита _____

(підпис)

Анотація

В бакалаврському проекті було створено додаток, що використовує машинне навчання.

У роботі проведено аналіз існуючих рішень, що вирішують дану проблему, розглянуто переваги та недоліки кожної з систем. Для розв’язання задачі було розглянуто декілька алгоритмів машинного навчання та за допомогою google colab була натренована модель. Наступним завданням було покращення сервісу за допомогою зміни структури нейронної мережі.

Annotation

Task of this bachelor project is creating of application that uses neural networks and machine learning.

In this thesis modern solutions were analyzed; they were compared. In order to solve task were implemented several algorithms of machine learning as platform for development was used google colab as cloud solution for machine learning. Also were provided several experiments in model structure in order to improve model`s accuracy.

ВІДОМІСТЬ ДО ДИПЛОМНОГО ПРОЕКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на диплом	2	
2	А4	*ІАЛЦ.467200.000 ВП*	Відомості проекту	1	
3	А4	*ІАЛЦ.467200.001 ТЗ*	Технічне завдання	4	
4	А4	ІАЛЦ.467200.002 ПЗ	Пояснювальна записка	50	
5	А4	ІАЛЦ.467200.003 Д1	Схема Структурна	1	
6	А4	ІАЛЦ.467200.004 Д2	Схема Функціональна	1	
7	А4	ІАЛЦ.467200.005 Д3	Схема принципова	1	
8	А4	*ІАЛЦ.467200.006 Д4*	Лістинг програми	4	

					ІАЛЦ. 467200.000 ВП		
Зм.		*№ документа*	*Підпис*	*Дата*			
Розробив		*Сміщенко Б.О*			Система виявлення вторгнень	*Літ.*	*Аркуш* 5 / *Аркушів* 1
Перевірив		Волокита А.М.					
Н. Контр.		*Симоненко В.П.*				*НТУУ «КПІ ім. Ігоря Сікорського» ФІОТ гр.*	
Затвердив							

ТЕХНІЧНЕ ЗАВДАННЯ

до дипломної роботи

освітньо-кваліфікаційного рівня «бакалавр»

на тему: «Система виявлення вторгнень у мережі»

Київ 2020

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання поширюється на курс «Система виявлення вторгнень у мережі». Область застосування: використання веб сервісами для забезпечення безпеки.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на кваліфікаційно-освітнього рівня «Інженерія програмного забезпечення комп'ютерних систем», затверджене кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний інститут ім. Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного проекту є розробка додатку з використанням машинного навчання та використання різних підходів та алгоритмів для тренування нейронних мереж та машинного навчання.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки є науково-технічна література з даного питання, публікації та патенти, що пов'язані з цією темою. Освітні статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розробляємого продукту

- Використання широкого спектру алгоритмів та архітектур машинного навчання.
- Безпека системи – відмовостійкість від можливих атак на систему.
- Підлаштування системи для кожного клієнта.

5.2. Вимоги до апаратної частини

- Операційна система MS Windows 7 service pack 2 або краще.
- Доступ до мережі Інтернет.

- Браузер.

5.3. Вимоги до апаратної частини

- Комп’ютер на базі процесора Amd/Intel
- Оперативної пам’яті 2 Гб

6. ЕТАПИ РОЗРОБКИ

Етап	**Дата**
Вивчення літератури	15.03.2020
Складання і узгодження технічного завдання	1.04.2020
Створення модулів системи	15.04.2020
Експерименти з підвищення точності	20.04.2020
Доопрацювання, виправлення помилок	6.05.2020
Оформлення документації дипломної роботи	8.05.2020

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломної роботи

освітньо-кваліфікаційного рівня «бакалавр»

на тему: «Система виявлення вторгнень»

					ІАЛЦ. 467200.002 ПЗ		
Зм.		*№ документа*	*Підпис*	*Дата*			
Розробив		*Сміщенко Б.О*			Система виявлення вторгнень	*Літ.*	*Аркуш* 10 / *Аркушів* 1
Перевірив		Волокита А.М.					
Н. Контр.		*Симоненко В.П.*				*НТУУ «КПІ ім. Ігоря Сікорського» ФІОТ гр.* ІП-62	
Затвердив		.					

ЗМІСТ

ВСТУП

Алгоритми для реалізації машинного навчання з’явились у 1970 роках, але технологія машинного навчання здобула свою популярність лише у 10-х роках 21 сторіччя через появу дешевих та ефективних рішень, що здатні паралельно проводити тисячі обчислень. Також розвиток хмарних технологій дозволив дослідженням з нейронних мереж, глибокого навчання, машинного навчання бути проведеним з будь-якого місця на планеті. Серед популярних хмарних рішень для науки про данні та машинного навчання можна назвати:

1. Google Collaboratory
2. Azure Notebooks
3. Jupiter Notebook
4. Kaggle

Машинне навчання використовують у сферах, де необхідно оброблювати велику кількість даних, що не є можливим для людини. Машине навчання – додаток, що надає системі можливість автоматично навчатись і покращуватись, завдяки досвіду без прямого програмування.

Серед сфер, які користуються додатками з машинним навчанням – динамічне обчислення ціни проїзду таксі Uber, Gmail використовує цю технологію для виявлення спаму. Також машинне навчання використовується у медицині для підвищення точності виявлення хвороб. Сфера розваг та торгівельні мережі використовують цю технологію для виявлення вподобань клієнта та підбір товару.

Поява мікросервісної архітектури та хмарних обчислень зробила можливим розгортання додатку віддалено та масштабування окремих сервісів додатку.

Проблема виявлення атак, що погрожують серверу або усій сітьовій архітектурі, постала з 1996. Донедавна використання нейронних мереж було надто дорогим через високу ціну обладнання та його малу ефективність. Поява

дешевих обчислювальних потужностей дозволила зробити використання нейронних мереж ефективним рішенням багатьох задач, за досить низьку ціну. Одночасно з цим, поширення веб технологій дозволило створювати комплексні додатки, що включають в себе модулі, що працюють ізольовано один від одного.

Об’єктом дипломної роботи є створення системи, здатної відстежувати вторгнення направлені на сервер ззовні. Вибір об’єкту дослідження обумовлено бажанням розширити навички використання штучного інтелекту, також бажанням вивчити засоби підвищення точності висновків нейронної мережі.

Предметом дослідження є створення додатку, здатного класифікувати запити на безпечні та небезпечні.

Метою роботи є:

1. Розгляд доцільності використання нейронних мереж для створення Intrusion prevention system.
2. Створення додатку, здатного класифікувати запита як безпечні та небезпечні.
3. Спроба створення частини додатку, що трансформує запит до необхідної форми.

Були поставлені наступні завдання:

1. Огляд існуючих рішень
2. Реалізація Intrusion Prevention System
3. Перегляд засобів підвищення точності нейронних мереж

Дипломна робота складається зі вступу, чотирьох розділів, висновку, та додатку до проекту. У першому розділі розглянуто існуючі системи, їх історія, платформи для розробки машинного навчання, види машинного навчання, їх відмінності. У другому розділі розглянуто відмінності між мовами програмування у сфері машинного навчання, фреймворк Microsoft ML та tensorflow на мовах c# та python відповідно.

У третьому розділі описаний процес створення система та деталі реалізації. Розглянуто декілька архітектур для реалізації додатку з машинним навчанням. Наведено приклад використання unsupervised та supervised learning, створення додатку з навчанням без учителя та описаний процес підбору необхідних параметрів та їх вплив на фінальну точність. Розглянуто перехід від задачі багатокласової класифікації до двійкової класифікації. Наведено переваги та недоліки використання кожного з підходів та можливі засоби вирішення складнощів, викликаних недоліками.

У четвертому розділі показані намагання підвищити точність нейронної мережі за допомогою зміні функції активації та модифікації структури нейронної мережі. Також у четвертому розділі наведені підходи до встановлення створеної системи для різних топологій. Наведено пропозиції модифікації існуючих систем та переваги встановлення IPS.

РОЗДІЛ 1 СИСТЕМИ ВИЯВЛЕННЯ ВТОРГНЕНЬ

В першому розділі розглянуто види систем протидії вторгненням.

На сьогоднішній момент, система IPS та IDS широко використовуються для автоматичного контролю за трафіком, що надходить від клієнтів. Засоби якими досягаються такі цілі поділяються на підвиди, такі як IPS з функцією сканеру – така програма шукає вразливості у системі захисту та вказує на них. Є активні системи моніторингу, що визначають аномалії у поведінці користувачів та вживають належних заходів у напівавтоматичному режимі. Далі у роботі будуть розглянуті приклади IPS та IDS та їх відмінності.

1.1 Архітектура та доцільність використання IDS

У наш час, IDS автоматично та у реальному часі будують профіль користувача, що дозволить визначити звичайну поведінку для кожного користувача. Система помічає неавторизований доступ або людиною, або комп’ютером через порівняння поведінки користувача з профілем користувача. Коли система помічає неавторизовані дії, вона викликає контрольну функцію, що автоматично виконує необхідні дії для відповіді на подію.

Профіль користувача динамічно оновлюється, починаючи зі спроб користувача увійти в систему і після процесу входу профіль користувача оновлюється. Завдяки такому підходу, коли профіль користувача порівнюється з динамічно оновлюваним записом, вдалося зменшити кількість помилкових тривог.

Також система включає в себе функцію логування, функцію сканування портів та функцію моніторінгу сесії. [9]

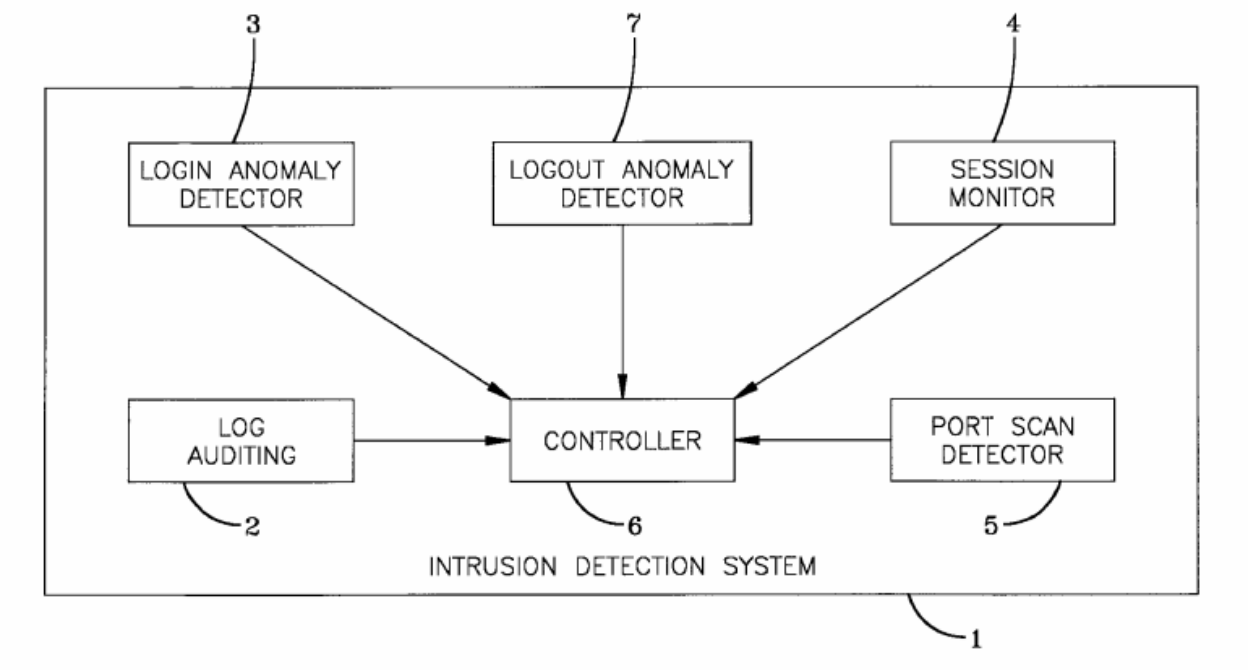


Рисунок 1.1. Ілюстрація патенту на основні модулі IDS.[9]

Вторгнення – це неавторизований доступ чи спроба здобути такий доступ також, неавторизована діяльність у комп'ютері чи інформаційній системі. Використання IDS дозволяє значно збільшити загальну безпеку комп'ютерної системи.

В основному, дані для роботи IDS збираються та скорочується автоматично, але аналіз даних все ще проводиться вручну. Для вирішення цієї задачі використовуються декілька підходів:

- Профілювання – екстраполяція даних за певним критерієм
- Розпізнавання шаблонів – автоматичний пошук закономірностей у даних.

Серед задач, що вирішуються вручну є визначення нормальної поведінки користувача, додатку чи системи. Нормальна поведінка потім використовується для створення правил. Значні відхилення від встановлених правил відповідає аномальній поведінці

Деякі IDS, основані на пошуку аномалій, шукають статистично аномальну поведінку, тобто поведінку, що здається незвичайною у порівнянні з поведінкою інших користувачів. Єдиним недоліком системи пошуку аномалій є вразливість до хибно позитивного та хибно негативного результату через те,

що правила, визначені заздалегідь, не специфічні для поведінки кожного користувача.

Хибно позитивний результат трапляється у випадках, коли IDS ідентифікує подію як вторгнення, коли воно не трапилось.

Хибно негативний результат трапляється у випадках, коли IDS не може ідентифікувати вторгнення або під час вторгнення, або після того, як воно відбулось.

Деякі системи виявлення вторгнень використовують експертні системи, які керуються з закодованої бази правил для моніторингу дотримання політики. Експертна система застосовує правила, щоб переконатись, що всі користувачі працюють у межах своїх привілеїв. Навіть у експертній системі зазвичай закодовані правила породжене профілюванням аномальної поведінки, на основі яких потім будується системи. Це означає, що експертна система виявлення вторгнень страждає від таких проблем, як помилково позитиві та помилкові негативні спрацювання.

Існують також інші підходи до створення IDS:

- Пасивні функції моніторингу даних – система не здатна на якісь дії, а лише аналізує увесь трафік, що проходить крізь неї.
- Сканери – системи, що постійно шукають вразливості як у коді так і у обладнанні. Такий підхід має свої недоліки, як наприклад здатність системі розпізнавати лише загрози, що вже відомі, тобто нездатність реагувати на новітні загрози.[https://patents.google.com/patent/US6405318B1/en]

Як висновок, використання IDS може бути ефективним лише у випадку прямого контролю від оператора та прийняття ним рішень для усунення проблем, знайдених завдяки IDS. Тому використання IDS доцільне у комбінації з іншими засобами контролю трафіку у мережі.

1.2. Архітектура та доцільність використання IPS

Проблеми, що має вирішує IPS, така сама як і IDS, тобто вирішення проблем здобуття неавторизованого доступу до профілю користувача, серверу та комп'ютерної системи.

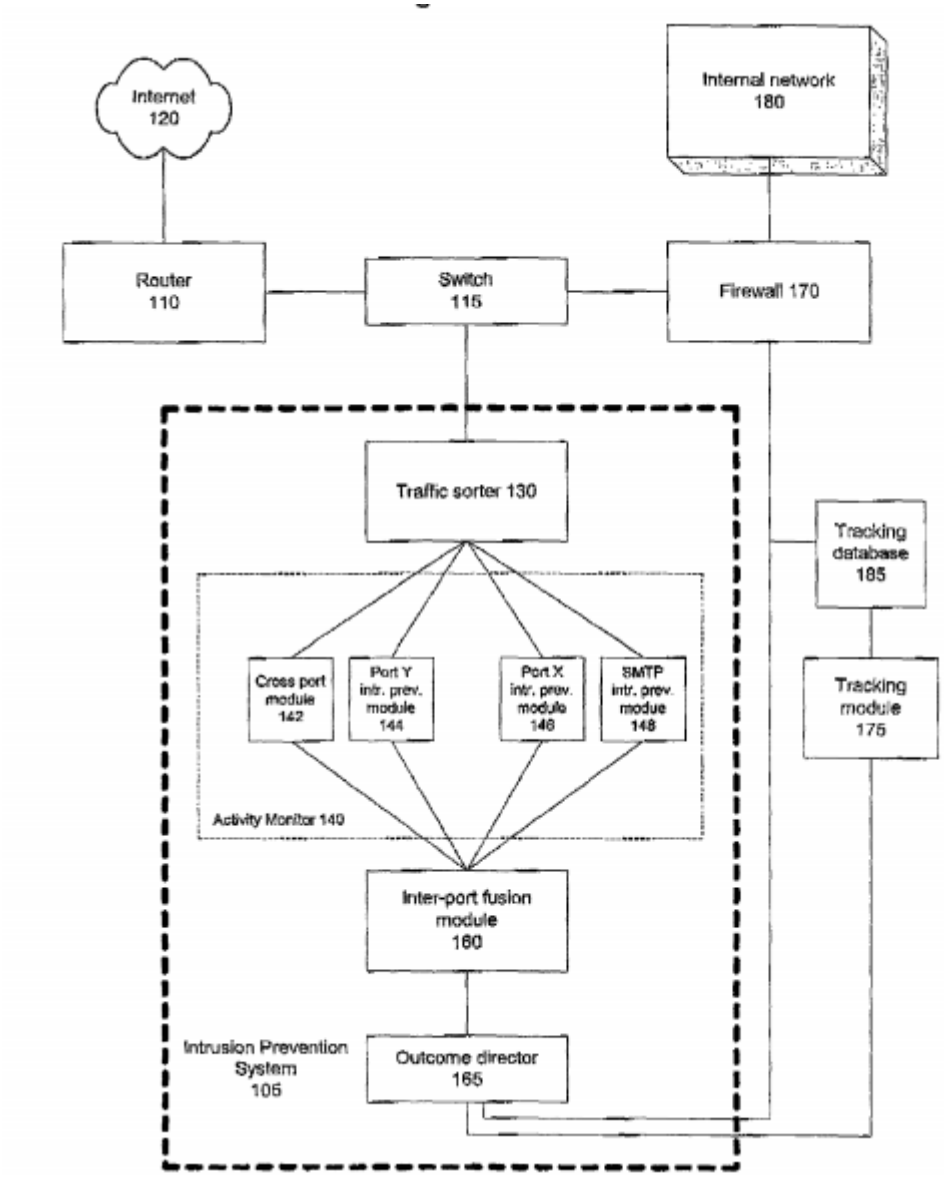


Рисунок 1.2. Загальна архітектура IPS.[10]

Вторгнення можуть розрізнятись від невинних користувачів, що випадково отримав доступ до інформації, до висококваліфікованої людини, що умисно здобула доступ до захищеної інформації з метою нанести втрат системі.

Відмінність від IDS полягає у здатності змінювати налаштування файерволу та автоматично реагувати на атаки та відновлювати систему після них.

Види IPS:

- Співпадіння шаблонів – система наглядає за поведінкою зловмисника та шукає шаблони його поведінки. Завдяки цьому можливі широкомасштабні виявлення раніше застосованих атак.
- Пошук аномалій – новий підхід, заснований на створенні профілю користувача, завдяки використанню нейронних мереж та самонавчанню, створюється профіль статистично «нормального» користувача та його взаємодії з мережею. Якщо поведінка користувача відрізняється від норми, то система автоматично блокує такий профіль або такий IP. Такий підхід дозволяє реагувати на новітні загрози. Але існують і недоліки такого підходу:
    - Як незвична активність можуть бути ідентифіковані цілком звичайні дії, що призведе до блокування користувача.
    - Неможливість легко підлаштовувати систему до використання IPS, тобто необхідний довгий період налаштувань системи та тренування її на основі конкретних користувачів.

Тому використання або IPS, або IDS поодинці не є ефективним кроком. В той час, коли IDS пропонує лише сповіщення, тобто основні дії мають бути виконані людиною, що сповільнює реакцію системи на загрози. А IPS автоматично робить необхідні кроки для захисту, але страждають користувачі.

Тобто ефективним буду одночасне використання IDS для створення профілю користувача, а IPS для сповіщення та необхідних дій, але з можливістю відновлення попереднього стану системи.

1.3. Приклади IPS та IDS

Розглянемо сучасні найбільш розповсюджені системи IPS/IDS.

1. Splunk
2. Sagan
3. OSSEC
4. Open WIPS-NG
5. Fail2Ban

6. Zeek

1.3.1. Приклади IDS

SPLUNK

Один з прикладів Intrusion Prevention System є Splunk.

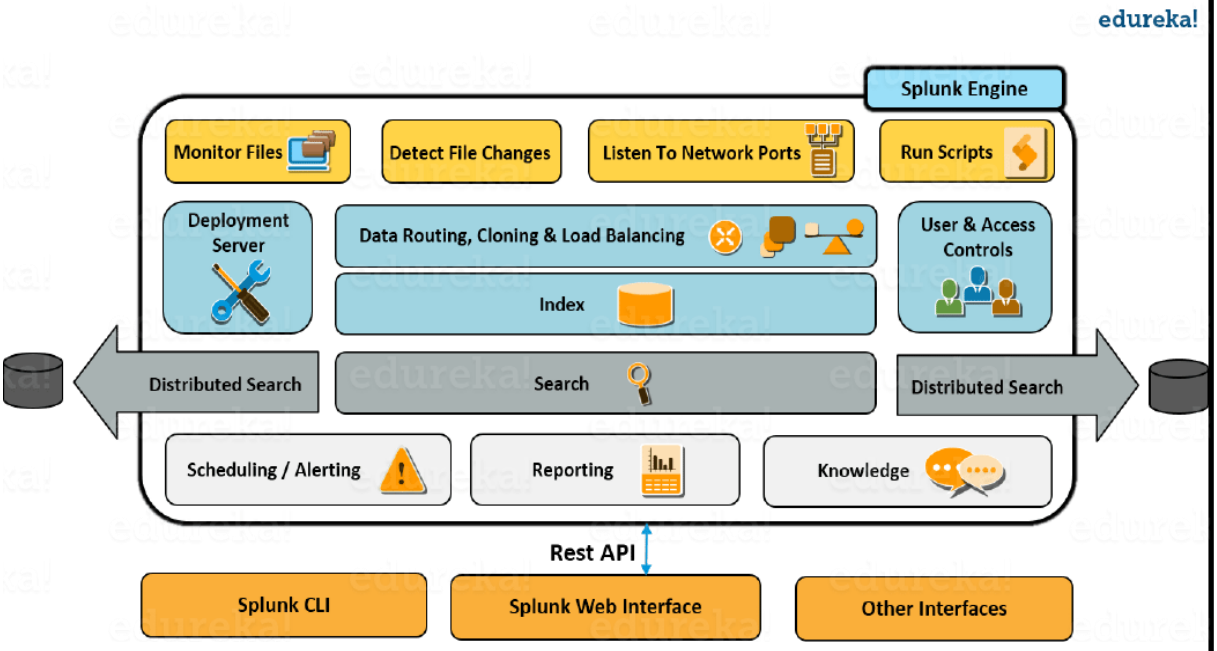


Рисунок 1.3 Структура Splunk[11]

Структура IPS Splunk наведена на малюнку 1.3. Опишемо основні можливості детальніше.

- Існує можливість автономізувати обробку файлів та запитів завдяки використанню скриптів.
- Створений балансувальник навантаження, який дозволяє, завдяки клонуванню даних, зберегти інформацію, навіть за втрати одного з потоків.
- Дані, що надходять ззовні, індексуються, що прискорює доступ до них та дозволяю опрацьовувати цілі категорії за єдиним зразком.
- Також існує можливість створення сповіщень, пов’язаних з певними типами даних.
- Додана можливість обробляти дані, використовуючи існуючі шаблони.

Як і інші системи IPS / IDS, цей приклад має декілька недоліків, наприклад, вразливість до новітніх загроз, яка однак може бути вирішена завдяки використанню користувацьких скриптів. Іншим недоліком є необхідність використання оператора, що буде контролювати роботу програми та за необхідності коригувати її.

OSSEC

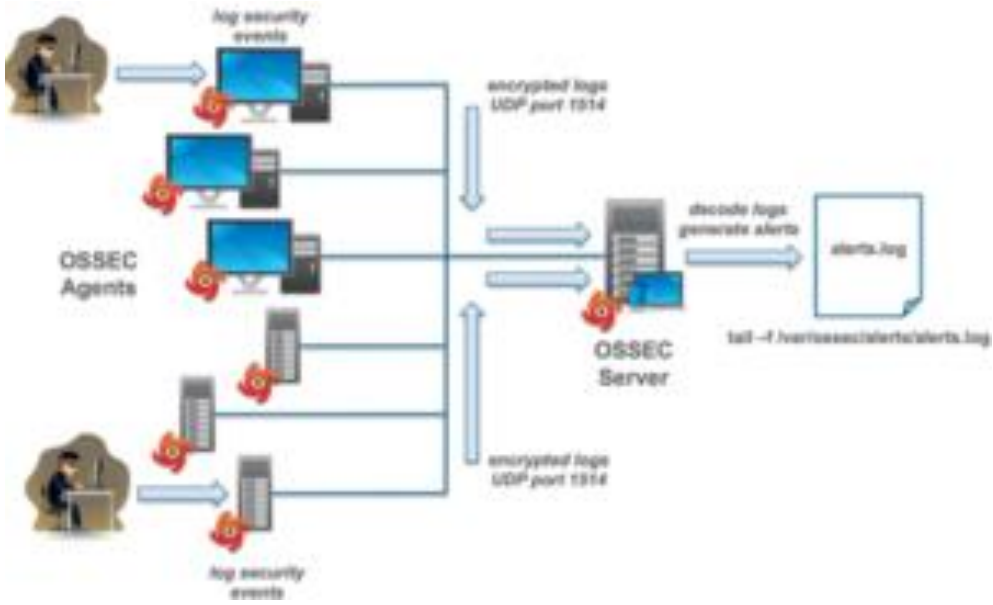


Рисунок 1.4. Структура OSSEC [12]

Структура OSSEC наведена на малюнку 1.4. Опишемо основні риси цього додатку детальніше.

- Відкрита реалізація, тобто кожна людина може додати необхідний функціонал.
- Аналіз логів.
- Перевірка цілісності файлів
- Для Windows можлива перевірка цілісності регістрів
- Сповіщення адміністратора
- Автоматично блокує брутфорс-атаки, через протоколи ftp, http,ssh

OSSEC використовує клієнт-серверну архітектуру, має центрального менеджера для моніторингу та отримання даних з агентів.

Менеджер – центральна частина OSSEC, що зберігає базу даних цілісності файлів, логи, події та записи аудиту системи. Усі правила та конфігурації, що дозволяє просто масштабувати кількість агентів.

Агент – програма або колекція програм, встановлені для моніторингу за системою. Агент збирає інформацію та передає її менеджеру для аналізу та пошуку зав'язків. Правила для пошуку вторгнень знаходяться у менеджера, уся обробка даних проходить у менеджера. Агент не здатен посилати сповіщеня до клієнта, він лише передає та збирає необроблені дані.

1.3.2. Приклади IPS

Fail2Ban

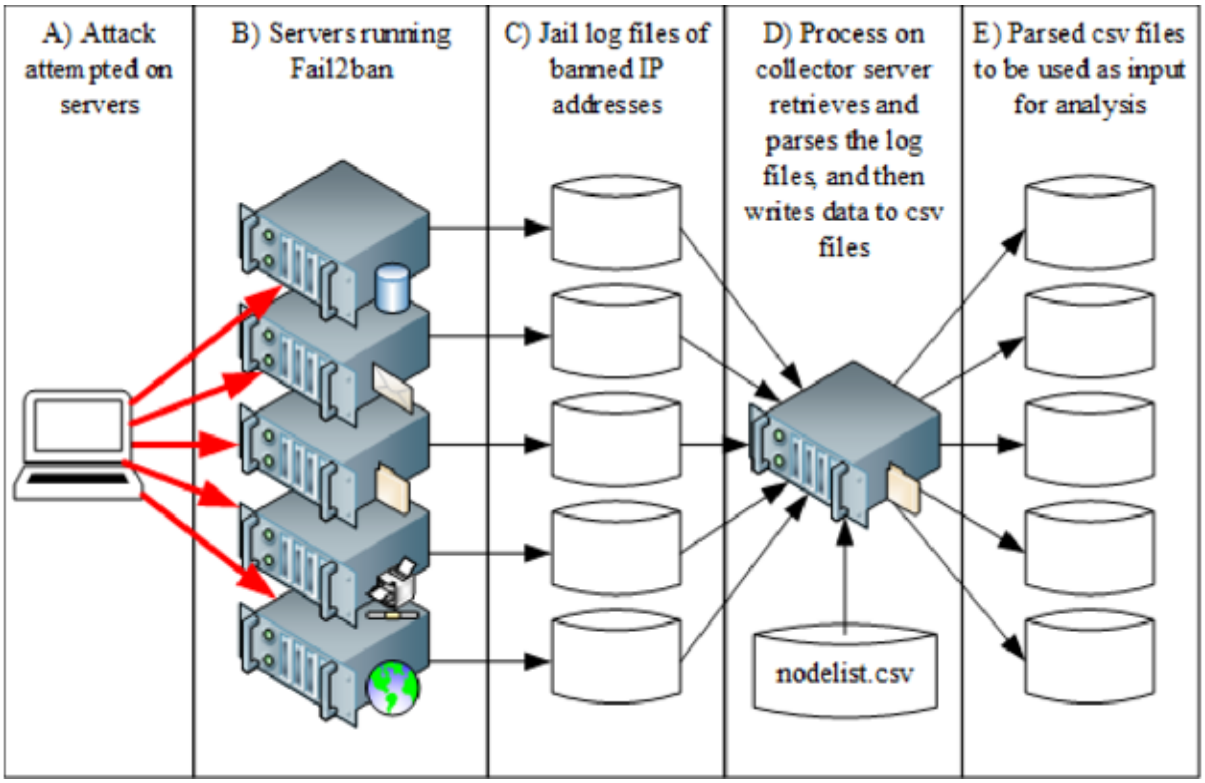


Рисунок 1.5. Структура Fail2Ban[13]

Структура Fail2Ban наведена на малюнку 1.5. Наведемо основні риси цього продукту:

- Клієнт-серверна архітектура
- Можливість модифікувати файервол
- Може відкидати підозрілі та небезпечні пакети
- Може виконувати команди на мові python3

- Логування до баз даних
- Можливість блокування деяких IP адрес

Fail2Ban сканує файли логів та блокує IP, що виглядають небезпечно, наприклад забагато спроб ввести коректний пароль, пошук вразливостей. Також модифікує файервол для блокування майбутніх загроз з цього IP.

Zeek

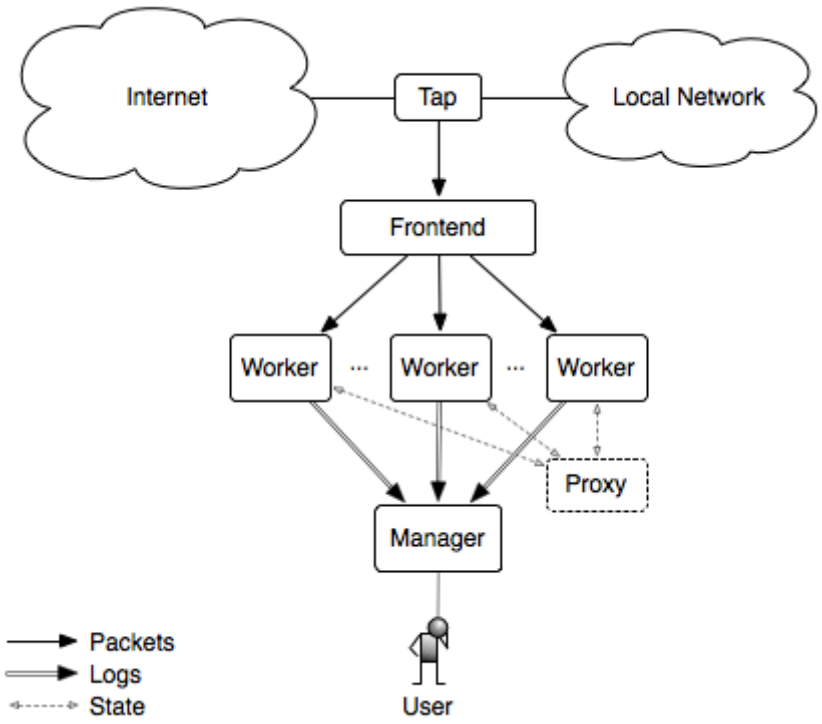


Рисунок 1.6. Структура Zeek[14]

На рисунку 1.6. наведена архітектура додатку Zeek, розглянемо основні компоненти архітектури:

Tap – механізм, що розділяє вхідний потік даних для того, щоб зробити копію для майбутнього аналізу.

Frontend – частина, що розподіляє вхідні дані на декілька потоків.

Manager – додаток, що отримує логує повідомлення та отримає сповіщення від інших вузлів. Результатом праці є створення єдиного журналу, що за рахунок інструментів менеджера буде проаналізовано.

Worker – частина архітектури, що аналізує вхідний трафік у пошуках загроз для серверу клієнта. Формує звіт та відправляє його менеджеру.

Основні риси цієї системи:

- Простота у масштабуванні системи
- Пасивний аналіз трафіку
- Підтримка власної скриптової мови
- Підтримка усіх можливих протоколів рівня додатку(Application layer)
- Підтримка IPv6

Як можна побачити з архітектури, система опрацьовує дані незалежно від того, чи погрожують вони системі. Це призводить до зниження ефективності попередження атак, але робить можливим відновлення послідовності запитів під час атаки. Тобто, доцільним є використання такої системи у комбінації з іншою IPS / IDS, де Zeek створюватиме правила та буде корисним інструментом для аналізу запитів вручну, а інша система буде працювати вже з готовим рішенням у автоматичному режимі.

1.4. Порівняльний аналіз систем IDS та IPS

Розгляне основні переваги та недоліки систем, які розглянуті вище. Відмінності між Intrusion Prevention System(IPS) та Intrusion Detection System(IDS).

Таблиця 1.1. Різниця між IDS та IPS

Функціонал	IDS	IPS
Аналізує інформацію всередині мережі	+	+
Сповіщення адміністратора	+	+
Має можливість блокувати підозрілі запити	+	+

Функціонал	IDS	IPS
Може модифікувати фаєрвол	-	+
Відновлює систему після атаки	-	+

Як можна побачити, IPS має більший функціонал, та є не консультаційним додатком, а таким, який сам може модифікувати елементи мережі, як наприклад, відновлення після атак, модифікація файерволу, перевірка цілісності файлів, пошук вразливостей, у той час як IDS лише фіксують трафік, що проходить мережею та сповіщає адміністратора.

Таблиця 1.2 Порівняння розглянутих систем

Наявний функціонал	SPLUNK	OSSEC	Fail2Ban	Zeek
Аналізує інформацію всередині мережі	+	+	+	+
Підтримка власної скриптової мови	-	-	+	-
Простота у масштабуванні системи	-	-	+	+
Сповіщення адміністратора	+	+	+	+
Аналіз логів	+	+	+	+
Робота з базами даних	+	+	+	+
Пошук вразливостей у мережі	-	-	-	+
Можливість модифікації мережі	-	-	+	-

Як можна побачити з таблиці 1.2. кожна з систем вирішує одне завдання – забезпечення безпеки мережі, але різними шляхами, наприклад, Zeek дозволяє пошук вразливостей, у той час як жодна інша не має такого функціоналу. Не дивлячись на схожість функціоналу SPLUNK та OSSEC,

1.5. Типи машинного навчання.

Машинне навчання розподіляється на 3 основні категорії – навчання з вчителем, навчання без вчителя, підкріплене навчання.

Навчання з вчителем – вид машинного навчання, коли система має вектор прикладів та вектор значень, система змінює коефіцієнт кожного значення у

векторі для підвищення точності висновку системи. Такий тип нейронної мережі використовується у , наприклад, системі прогнозування висновку лікаря. Популярні мови програмування та бібліотеки, що підтримують такий вид машинного навчання – **python/tensorflow, c#/MicrosoftML**.

Навчання без вчителя - вид машинного навчання, коли система має лише вектор даних та має якимось чином їх класифікувати та самостійно знайти зв'язки між даними. Такий тип нейронної мережі використовується у Netflix для вибору фільму, якій вірогідно вам сподобається. Популярні мови програмування та бібліотеки, що підтримують такий вид машинного навчання – **python/scikit-learn, c#/MicrosoftML**.

Система з підкріпленим навчанням – вид машинного навчання, коли система винагороджує мережу за деякі рухи. Такий тип нейронної мережі використовується у іграх для вибору дій бота.

Також популярним рішенням є комбінація декількох типів машинного навчання, як наприклад, на початку вхідні дані кластерізуються за методом K-NN. Потім, результат кластеризації застосовується у нейронній мережі.

Серед найбільш популярних алгоритмів для реалізації кожного з підходів можна назвати:

1. Decision trees – структура являє собою дерево, де на листях знаходяться цільові значення, а на ребрах – записані атрибути, що визначають переходи. На внутрішніх вузлах записані атрибути, що являють собою логічні вирази.
2. K-Nearest Neighbors – використовується для класифікації та регресії, для використання необхідно визначити функцію відстані. При використовувати для класифікації об'єктів, клас елементу визначається за k найближчими сусідами. При використанні для регресії об'єкту присвоюється середнє значення k елементів.
3. Support Vector Machine – може використовуватись як для класифікації, так і для регресії. Алгоритм розділяє простір даних на кілька паралельних гіперплощин.

Для захисту клієнт-серверної архітектури необхідно відстежувати запити які надходять до серверу.

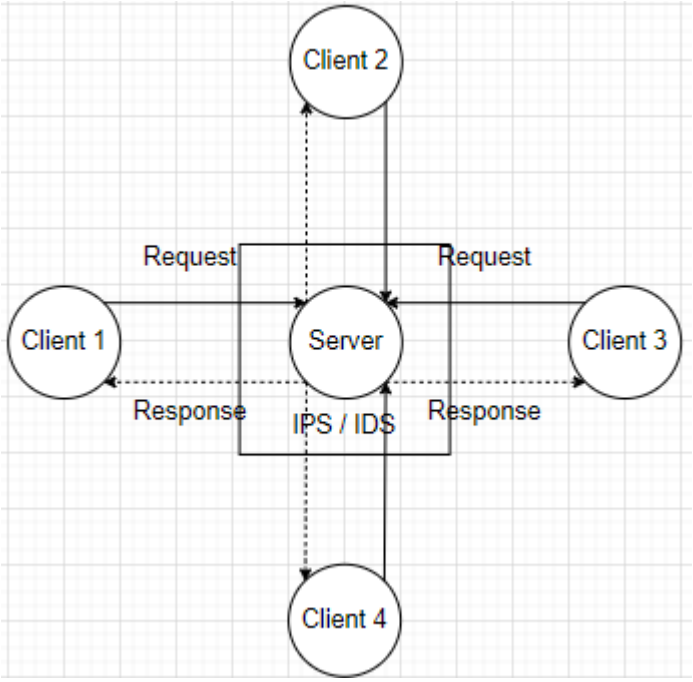


Рисунок 1.1 Ілюстрація клієнт- серверної архітектури

Як можна побачити на рисунку 1.1 IDS або IPS, контролює усю мережеву активність яка досягає сервера а також створену ним.

Використання клієнт-серверної архітектури шир око розповсюджено у промислових додатках, але як і будь-яка інша архітектура, клієнт-серверна має свої вразливості.

З клієнтської сторони, загроза для серверу може полягати у програмному забезпеченні, що має доступ до створення або модифікації запитів клієнта без будь-якого сповіщення. Також, можливе використання зловмисником мережевих сокетів та встановлення неавторизованого зв'язку.

Зі сторони серверу, загрози набагато небезпечніші для додатку та даних користувачів, таким чином небезпека після втрати контролю може бути набагато більшою. Найбільш популярним типи атак:

1. SQL injections
2. Broken authentication
3. Vulnerabilities caused by invalidated forwards and redirects

4. Usage of outdated encrypting algorithms and vulnerabilities in up to date.

Наприклад, зловмисник може перехопити запит клієнта і якщо запит неправильно зашифровано чи, коли зловмисник має доступ до пристрою клієнта та модифікує його запит.

1.6. Види баз даних для зберігання інформації

У наш час, основними архітектурами баз даних є нереляційні та реляційни бази даних. Розглянемо значущі відмінності між ними.

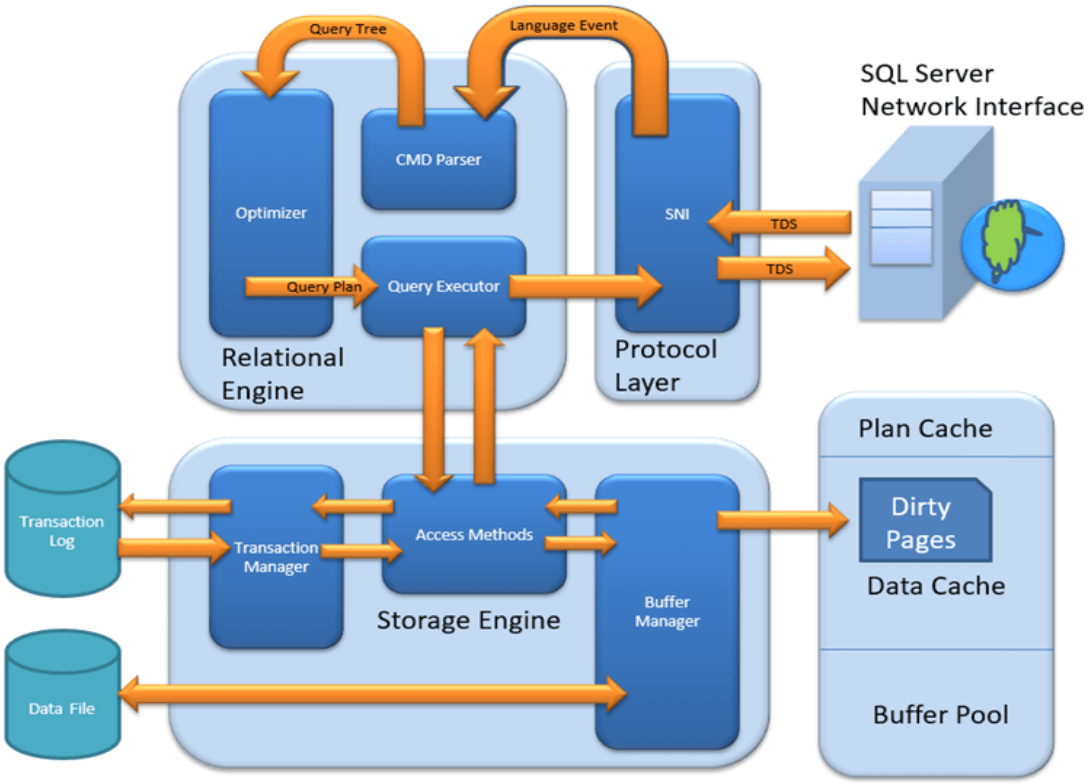


Рисунок 1.7. Структура MS Sql Server[15]

При використанні реляційних баз даних, будуть мати місце наступні риси.

Таблиця 1.2. недоліки та переваги реляційних БД

Риса	Так/ні
Можливість використовувати скриптову мову	+
Чітка форма зберігання даних	+
Простота реплікації даних	-
Економія місця при зберіганні даних	+

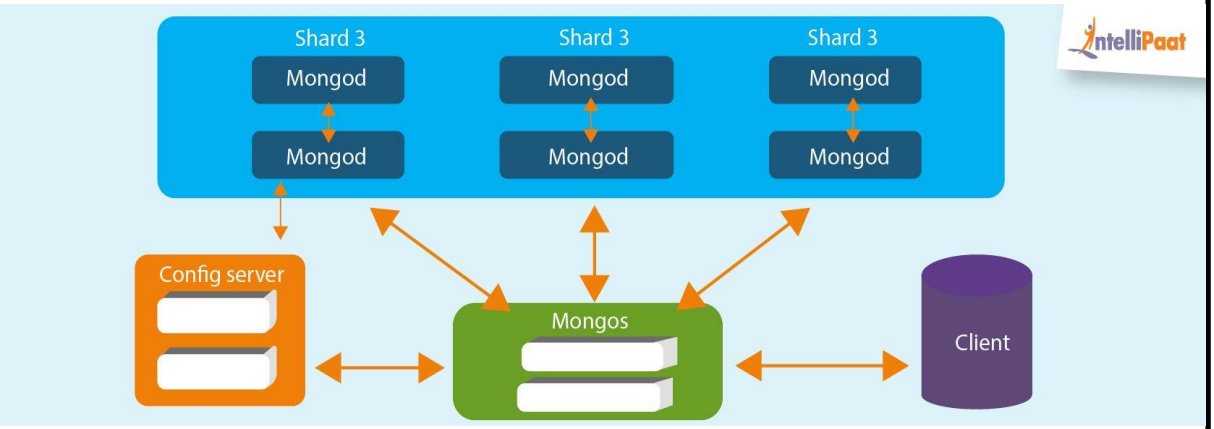


Рисунок 1.8. Архітектура Mongodb

Таблиця 1.3. Риси NoSql databases

Риса	Так/ні
Використання скриптової мови	-
Чітка форма зберігання даних	-
Простота реплікації даних	+
Дороговизна зберігання даних	+

Як можна побачити, простота зберігання неструктурованих даних робить використання Nosql db надзвичайно вигідним.

Але для структурованих даних вигідніше використовувати Sql db, наприклад для зберігання перетворених запитів користувачів до необхідного формату.

ВИСНОВКИ РОЗДІЛУ 1

У першому розділі було розглянуті види машинного навчання, а саме – навчання з вчителем, навчання без вчителя та підкріплене навчання.

Були розглянуті існуючі рішення IPS/IDS, також наведені відмінності між ними.

Були розглянуті недоліки клієнт-серверної архітектури та найбільш популярні атаки. Також були розглянуті можливі атаки та відмінність між загрозами зі сторони клієнта та зі сторони сервера.

Таким чином, актуальною є задача розробки системи попередження і розпізнавання вторгнень, яка дозволить автоматично аналізувати трафік всередині мережі, сповіщати адміністратора про підозрілу діяльність користувачів. Система матиме можливість блокувати підозрілий трафік та підозрілу поведінку користувача.

Використання IPS окремо від інших систем, у автоматичному режимі, може викликати численні помилки у роботі сервісу, наприклад користувач може змінити пристрій, за допомогою якого він використовує сервіс, а система може визначити його поведінку як аномальну і заблокувати користувача. Тому доцільно використовувати інший сервіс, що контролюватиме блокування користувачів та надаватиме можливість відновити акаунт після блокування.

Також система може не мати права на блокування користувачів, а лише видавати рекомендації для адміністраторів, але такий підхід не дозволить швидко реагувати на загрози для сервісу.

РОЗДІЛ 2 ВИБІР ІНСТРУМЕНТІВ ДЛЯ РОЗРОБКИ

2.1. Вимоги до мов програмування та бібліотек

Для реалізації диплому розглядаються дві мови програмування – c# та python, та бібліотеки для обробки вхідних даних.

Для початку, використання мови c# має наступні ознаки – статична типізация, тобто неможливість зміну типів у рантаймі, а також неможливість вписувати об’єкт до невідповідного класу.

2.1.1. Бібліотека Microsoft.ML

Для створення нейронної мережі була використана бібліотека Microsoft ML. Серед переваг використання цієї бібліотеки та такого підходу – широкий вибір типів задач, що можуть бути вирішені. Але існують і недоліки – максимальна довжина файлу обмежена 100 000 строк. Іншою особливістю бібліотеки є автоматичне визначення найкращого алгоритму та закрита реалізація нейронної мережі, тобто структура мережі незмінна та невідома: вхідний шар відповідає розмірності вашого постачальника даних, а вихідний шар складається з 1 елемента.

Таблиця 2.1 Основні риси бібліотеки Microsoft.ML

Перелік підтримуваних можливостей	Так/ні
Увесь спектр алгоритмів машинного навчання з коробки	+
Автоматичне визначення найкращого алгоритму	+
Можливість змінити структуру нейронної мережі	-
Можливість змінити алгоритм машинного навчання	-
Автоматична генерація усього необхідно коду	+

Як показано в таблиці 2.1, такий спосіб створення програми, що використовує машинне навчання дозволяє суттєво спростити поріг входу, тобто будь-яка людина, незалежно від освіти, зможе використати такий інструмент для вирішення своїх задач. Єдиними вимогами до користувача є встановлена Visual Studio 2019 та знання мови програмування c# на базовому рівні. Після

використання модулю Microsoft.ML у середовищі visual studio буде створено наступні проекти, при виборі імені проекту “modelBuilder”:

- modelBuilderML.ConsoleApp – проект, що відповідає за створення та зберігання моделі (об’єкту, що відповідає за класифікацію вхідної інформації), також відповідає за створення пайплайну, тобто функції, що трансформує вхідні данні у числові відповідники. Також цей проект відповідальний за тренування, вираховування ефективності та реалізацію алгоритму машинного навчання.

Рисунок 2.1 Опис функцій з проекту modelBuilderML.ConsoleApp

- modelBuilderML.Model – проект, що відповідає за перетворення інформації з постачальника даних у об’єкти класу, також згенерованого у цьому проекті. Також у проекті знаходиться опис результуючого класу.

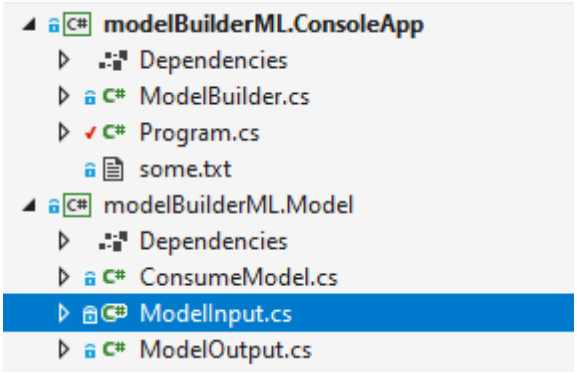


Рисунок 2.2 Структура проектів, створених за допомогою Microsoft.ML

Через неможливість впливати на структуру нейронної мережі та неможливість змінити алгоритм використання цієї бібліотеки є недоцільним.

2.1.2. Використання мови c#

Для реалізації цього проекту був використаний модуль “modelBuilderML.Model” який дозволяє зібрати інформацію з файлів формату .csv. А структура нейронної мережі та алгоритм активації був написаний на мові c#. Такий підхід дозволяє змінювати структуру нейронної мережі, змінювати функцію активації, також розмір вхідного файлу не мав обмежень на кількість значень.

Таблиця 2.2. Основні риси використання мови c#

Перелік підтримуваних можливостей	Так/ні
Увесь спектр алгоритмів машинного навчання, що знає програміст	+
Автоматичне визначення найкращого алгоритму	-
Можливість змінити структуру нейронної мережі	+
Можливість змінити алгоритм машинного навчання	+
Автоматична генерація усього необхідно коду	-

Як показано у таблиці 2.2. такий спосіб створення додатку з нейронною мережею серед переваг можна навести можливість налаштовувати структуру нейронної мережі, змінювати алгоритми активації. Вимоги до використання такого є досить високими, тобто людина має володіти такими навичками як:

- Значні знання мови c#
- Значні знання функцій активації та функцій розповсюдження помилок
- Математичні функції підрахування помилки

Тобто, такий спосіб створення нейронних мереж потребує значної кваліфікації розробника. Також підміна алгоритмів дуже затратна та кропітка.

```
foreach (var item in output)
{
    MaxSrcbytes =  MaxSrcbytes > item.Src_bytes ? MaxSrcbytes : item.Src_bytes;
    MaxDst_bytes = MaxDst_bytes > item.Dst_bytes ? MaxDst_bytes : item.Dst_bytes;
    MaxIs_guest_login = MaxIs_guest_login > item.Is_guest_login ? MaxIs_guest_login : item.Is_guest_login;
    MaxCount = MaxCount > item.Count ? MaxCount : item.Count;
    MaxSrv_diff_host_rate = MaxSrv_diff_host_rate > item.Srv_diff_host_rate ? MaxSrv_diff_host_rate : item.Srv_diff_host_rate;
    MaxDst_host_count = MaxDst_host_count > item.Dst_host_count ? MaxDst_host_count : item.Dst_host_count;
    MaxDst_host_srv_count = MaxDst_host_srv_count > item.Dst_host_srv_count ? MaxDst_host_srv_count : item.Dst_host_srv_count;
    MaxService = MaxService > item.Service ? MaxService : item.Service;
    MaxSrvCount = MaxSrvCount > item.Srv_count ? MaxSrvCount : item.Srv_count;
    MaxSerror_rate = MaxSerror_rate > item.Serror_rate ? MaxSerror_rate : item.Serror_rate;
    MaxFalg = MaxFalg > item.Flag ? MaxFalg : item.Flag;
    MaxDuration = MaxDuration > item.Duration ? MaxDuration : item.Duration;
    MaxProtocol_type = MaxProtocol_type > item.Protocol_type ? MaxProtocol_type : item.Protocol_type;
    MaxLand = MaxLand > item.Land ? MaxLand : item.Land;
    MaxWrong_fragment = MaxWrong_fragment > item.Wrong_fragment ? MaxWrong_fragment : item.Wrong_fragment;
    MaxUrgent = MaxUrgent > item.Urgent ? MaxUrgent : item.Urgent;
    MaxNum_failed_logins = MaxNum_failed_logins > item.Num_failed_logins ? MaxNum_failed_logins : item.Num_failed_logins;
    MaxLogged_in = MaxLogged_in > item.Logged_in ? MaxLogged_in : item.Logged_in;
    MaxNum_compromised = MaxNum_compromised > item.Num_compromised ? MaxNum_compromised : item.Num_compromised;
    MaxRoot_shell = MaxRoot_shell > item.Root_shell ? MaxRoot_shell : item.Root_shell;
    MaxHot = MaxHot > item.Hot ? MaxHot : item.Hot;
    MaxNum_acess_files = MaxNum_acess_files > item.Num_access_files ? MaxNum_acess_files : item.Num_access_files;
    MaxNum_Root = MaxNum_Root > item.Num_root ? MaxNum_Root : item.Num_root;
    MaxSu_attempted = MaxSu_attempted > item.Su_attempted ? MaxSu_attempted : item.Su_attempted;
    MaxFile_Creations = MaxFile_Creations > item.Num_file_creations ? MaxFile_Creations : item.Num_file_creations;
}
```

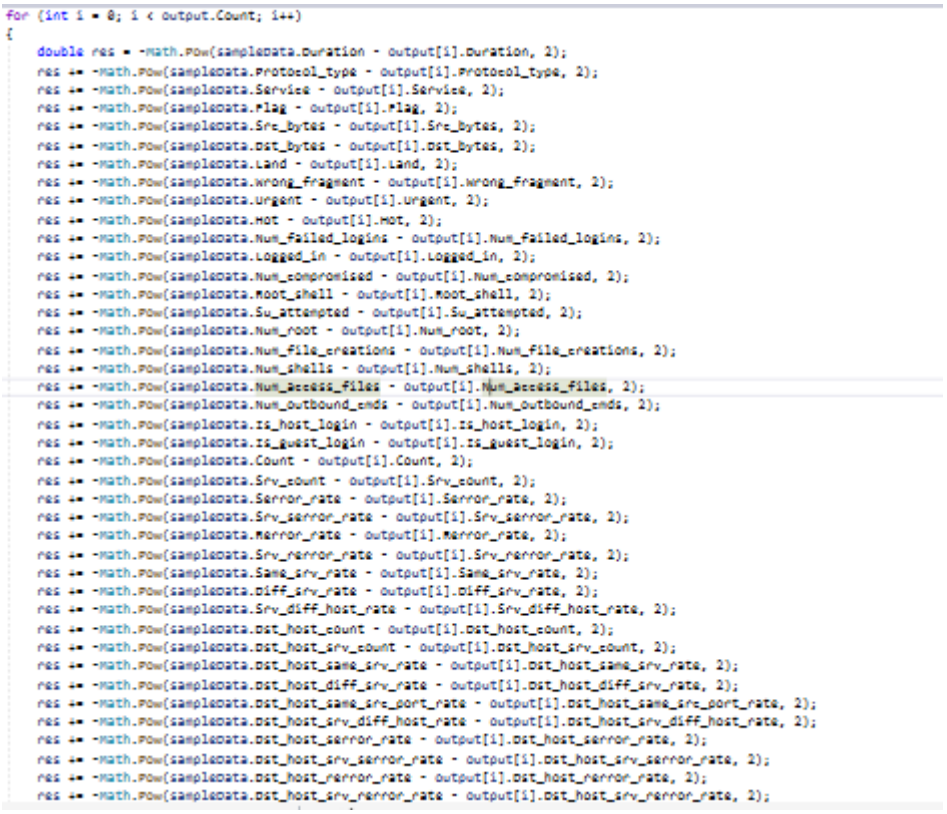
Рисунок 2.3. Пошук максимального елемента з кожного стовпця

Для перетворення даних до інтервалу [0..1], необхідно знайти максимальне значення кожного із стовпців. А потім розділити кожен елемент стовпця на максимальне значення, алгоритмічна складність такого процесу $O(n^2)$.

```
{
    sampleData.Src_bytes /= MaxSrcbytes;
    sampleData.Dst_bytes /= MaxDst_bytes;
    sampleData.Is_guest_login /= MaxIs_guest_login;
    sampleData.Count /= MaxCount;
    sampleData.Srv_diff_host_rate /= MaxSrv_diff_host_rate;
    sampleData.Dst_host_count /= MaxDst_host_count;
    sampleData.Dst_host_srv_count /= MaxDst_host_srv_count;
    sampleData.Service /= MaxService;
    sampleData.Srv_count /= MaxSrvCount;
    sampleData.Serror_rate /= MaxSerror_rate;
    sampleData.Flag /= MaxFalg;
    sampleData.Duration /= MaxDuration;
    sampleData.Protocol_type /= MaxProtocol_type;
    sampleData.Land /= MaxLand;
    sampleData.Wrong_fragment /= MaxWrong_fragment;
    sampleData.Urgent /= MaxUrgent;
    sampleData.Num_failed_logins /= MaxNum_failed_logins;
    sampleData.Logged_in /= MaxLogged_in;
    sampleData.Num_compromised /= MaxNum_compromised;
    sampleData.Root_shell /= MaxRoot_shell;
    sampleData.Hot /= MaxHot;
    sampleData.Num_access_files /= MaxNum_acess_files;
    sampleData.Num_root /= MaxNum_Root;
    sampleData.Num_file_creations /= MaxFile_Creations;

}
```

Рисунок 2.4. Перетворення данних



```
for (int i = 0; i < output.Count; i++)
{
    double res = -Math.Pow(sampleData.Duration - output[i].Duration, 2);
    res += -Math.Pow(sampleData.Protocol_type - output[i].Protocol_type, 2);
    res += -Math.Pow(sampleData.Service - output[i].Service, 2);
    res += -Math.Pow(sampleData.Flag - output[i].Flag, 2);
    res += -Math.Pow(sampleData.Src_bytes - output[i].Src_bytes, 2);
    res += -Math.Pow(sampleData.Dst_bytes - output[i].Dst_bytes, 2);
    res += -Math.Pow(sampleData.Land - output[i].Land, 2);
    res += -Math.Pow(sampleData.Wrong_fragment - output[i].Wrong_fragment, 2);
    res += -Math.Pow(sampleData.Urgent - output[i].Urgent, 2);
    res += -Math.Pow(sampleData.Hot - output[i].Hot, 2);
    res += -Math.Pow(sampleData.Num_failed_logins - output[i].Num_failed_logins, 2);
    res += -Math.Pow(sampleData.Logged_in - output[i].Logged_in, 2);
    res += -Math.Pow(sampleData.Num_compromised - output[i].Num_compromised, 2);
    res += -Math.Pow(sampleData.Root_shell - output[i].Root_shell, 2);
    res += -Math.Pow(sampleData.Su_attempted - output[i].Su_attempted, 2);
    res += -Math.Pow(sampleData.Num_root - output[i].Num_root, 2);
    res += -Math.Pow(sampleData.Num_file_creations - output[i].Num_file_creations, 2);
    res += -Math.Pow(sampleData.Num_shells - output[i].Num_shells, 2);
    res += -Math.Pow(sampleData.Num_access_files - output[i].Num_access_files, 2);
    res += -Math.Pow(sampleData.Num_outbound_cmds - output[i].Num_outbound_cmds, 2);
    res += -Math.Pow(sampleData.Is_host_login - output[i].Is_host_login, 2);
    res += -Math.Pow(sampleData.Is_guest_login - output[i].Is_guest_login, 2);
    res += -Math.Pow(sampleData.Count - output[i].Count, 2);
    res += -Math.Pow(sampleData.Srv_count - output[i].Srv_count, 2);
    res += -Math.Pow(sampleData.Serror_rate - output[i].Serror_rate, 2);
    res += -Math.Pow(sampleData.Srv_serror_rate - output[i].Srv_serror_rate, 2);
    res += -Math.Pow(sampleData.Rerror_rate - output[i].Rerror_rate, 2);
    res += -Math.Pow(sampleData.Srv_rerror_rate - output[i].Srv_rerror_rate, 2);
    res += -Math.Pow(sampleData.Same_srv_rate - output[i].Same_srv_rate, 2);
    res += -Math.Pow(sampleData.Diff_srv_rate - output[i].Diff_srv_rate, 2);
    res += -Math.Pow(sampleData.Srv_diff_host_rate - output[i].Srv_diff_host_rate, 2);
    res += -Math.Pow(sampleData.Dst_host_count - output[i].Dst_host_count, 2);
    res += -Math.Pow(sampleData.Dst_host_srv_count - output[i].Dst_host_srv_count, 2);
    res += -Math.Pow(sampleData.Dst_host_same_srv_rate - output[i].Dst_host_same_srv_rate, 2);
    res += -Math.Pow(sampleData.Dst_host_diff_srv_rate - output[i].Dst_host_diff_srv_rate, 2);
    res += -Math.Pow(sampleData.Dst_host_same_src_port_rate - output[i].Dst_host_same_src_port_rate, 2);
    res += -Math.Pow(sampleData.Dst_host_srv_diff_host_rate - output[i].Dst_host_srv_diff_host_rate, 2);
    res += -Math.Pow(sampleData.Dst_host_serror_rate - output[i].Dst_host_serror_rate, 2);
    res += -Math.Pow(sampleData.Dst_host_srv_serror_rate - output[i].Dst_host_srv_serror_rate, 2);
    res += -Math.Pow(sampleData.Dst_host_rerror_rate - output[i].Dst_host_rerror_rate, 2);
    res += -Math.Pow(sampleData.Dst_host_srv_rerror_rate - output[i].Dst_host_srv_rerror_rate, 2);
```

Рисунок 2.5 Реалізація функції Relu

Через те, що для коректої реалізації алгоритму машинного навчання є кропіткою роботою, широке розповсюдження такого підходу не виправдане, бо відладка та пошук помилок можуть займати багато часу.

2.1.3 Python + tensorflow

Для створення додатку з машинним навчанням на мові python використаємо бібліотеку tensorflow. Серед перваг такого методу – широкий вибір алгоритмів для вирішення задач, тобто бібліотека надає можливість реалізації усього спектру алгоритмів, але програміст має розроблювати структуру мережі.

Таблиця 2.3.

Перелік підтримуваних можливостей	Так/ні
Увесь спектр алгоритмів машинного навчання, що знає програміст	+
Можливість змінити структуру нейронної мережі	+
Автоматичне визначення найкращого алгоритму	-
Можливість змінити алгоритм машинного навчання	+
Автоматична генерація усього необхідно коду	-

```
for x in dataframe['protocol_type']:
    if x not in uniqueProtocolType:
        uniqueProtocolType.append(x)

for x in dataframe['service']:
    if x not in uniqueService:
      uniqueService.append(x)


for x in dataframe['flag']:
    if x not in uniqueFlag:
      uniqueFlag.append(x)

for x in dataframe['attackType']:
    if x not in uniqueFlag:
      uniqueAttackType.append(x)
#dataframe['attackType'] = [uniqueAttackType.index(item) for item in dataframe['attackType']]
dataframe['attackType'] = [(0 if item == 'normal' else 1) for item in dataframe['attackType']]
```

Рисунок 2.6. Заміна категоріальних даних у python.

Згідно з рисунком 2.6 спочатку ми збираємо усі можливі варіації текстових полів, а також змінюємо задачу з багатокласової класифікації на класифікацію на два класи, що дозволить спростити обчислення для мережі, та збільшити точність визначення класу.

```
model = tf.keras.Sequential([
    fearture_layer,
    layers.Dense(128, activation='relu'),
    layers.Dense(128, activation='relu'),
    layers.Dense(1, activation='sigmoid')
    #layers.Dense(1, activation='linear', input_dim = 22)
])
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'],
              run_eagerly=True)

print(train.dtypes)
model.fit(train_ds,
          validation_data=val_ds,
          epochs=3)

loss, accuracy = model.evaluate(test_ds)
print("Accuracy", accuracy)
```

Рисунок 2.7. Визначення структури нейронної мережі

Згідно з рисунком 2.7. визначаємо структуру нейронної мережі, що складається з трьох шарів – 1 – 128-128. Як функцію активації візьмемо relu та sigmoid

2.2 MS sql Server як засіб збереження даних

Для збереження трансформованих запитів використаємо sql базу даних через їх строгу формалізацію. Також необхідно спроектувати систему передбачивши посаду адміністратора, що зможе модифікувати таблицю з трансформованими запитами. Також необхідно додати безпекові профілі користувачів.

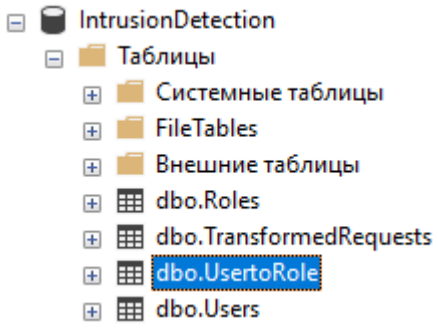


Рисунок 2.8 База даних, використана у проекті

Список таблиць у базі даних.

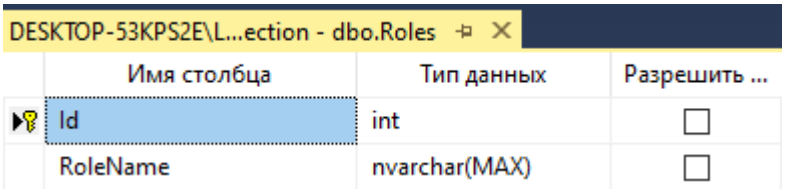


Рисунок 2.9. таблиця, що описує Ролі у проекті. Серед яких Адміністратор та звичайний користувач

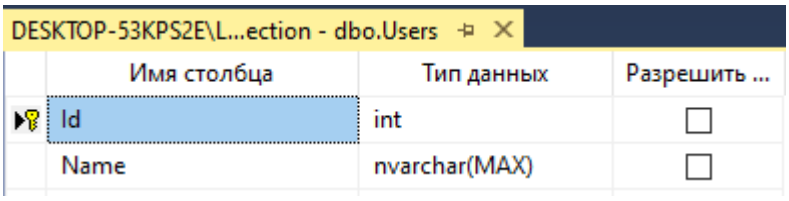


Рисунок 2.10. таблиця, що описує Користувачів у проекті

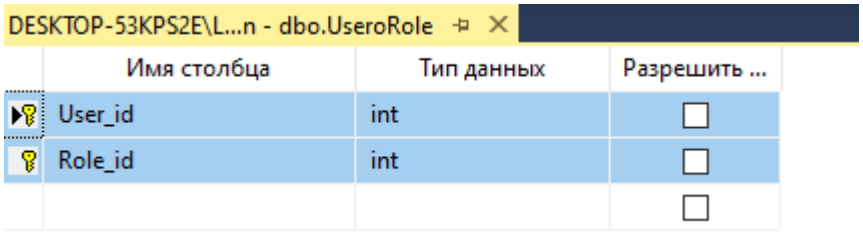


Рисунок 2.11. таблиця, що описує відношення між користувачами та їх ролями у проекті.

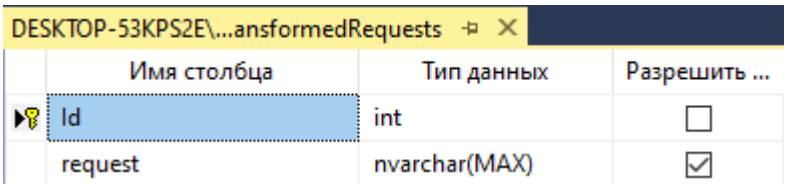


Рисунок 2.12 таблиця, що описує зберігання трансформованих запитів.

Згідно з малюнками 2.9 – 2.12 описана структура Бази даних, відповідальної за зберігання запитів та визначення ролей кожного користувача системи, що надає йому можливість додавати нові елементи до бази даних або нових користувачів та визначати їх права.

Використання Sql баз даних дозволяє структурувати дані, що надходять до них, також чисельні оптимізації кожної окремої СУБД дозволяють зберігати одні і ті самі дані меншим об’ємом ніж при використанні No Sql баз даних.

ВИСНОВКИ РОЗДІЛУ 2

У другому розділі були розлянуті можливі технології та засоби для реалізації додатку з використанням нейронних мереж. Наведені можливі переваги та недоліки кожного з методів.

Розглянутий засіб зберігання даних – MS Sql Server.

Таким чином для реалізації проекту була обрана мова python та бібліотека tensorflow, як найгнучкіший підхід до розробки додатку.

Також такий підхід дозволить ознайомитись з новими технологіям машинного навчання та застосувати широкий спектр технологій та підходів.

РОЗДІЛ 3.РОЗРОБКА СИСТЕМИ ВИЯВЛЕННЯ ВТОРГНЕНЬ

3.1 Задача виявлення вторгнень

Для створення додатку необхідні дані, які візьмемо з бібліотеки NSL-KDD, кожен рядок бібліотеки є вектором з 41 полем $\{X_1, X_2, X_3, X_4, .., X_i, .., X_m\}$. Кожен вектор складається як з числових так і текстових значень. Кожен вектор X_i відповідає значенню Y_i це означає, що файл складається з вектору $\{Y_1, Y_2, Y_3, Y_4, .., Y_i, .., Y_m\}$, де і – номер запиту всередині файлу, а m – кількість елементів усередині файлу. Однією з проблем цього датасету є неоднорідність кількості елементів кожного класу. Це призводить до зниження точності моделі майже до нуля.

```
0,tcp,ftp_data,SF,491,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0,0,0,0,1,0,0,150,25,0.17,0.03,0.17,0,0,0,0.05,0,normal,20
0,udp,other,SF,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,13,1,0,0,0,0,0.08,0.15,0,255,1,0,0.6,0.88,0,0,0,0,normal,15
0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,123,6,1,1,0,0,0.05,0.07,0,255,26,0.1,0.05,0,0,1,1,0,0,neptune,19
0,tcp,http,SF,232,8153,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,5,5,0.2,0.2,0,0,1,0,0,30,255,1,0,0.03,0.04,0.03,0.01,0,0.01,normal,21
0,tcp,http,SF,199,420,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,30,32,0,0,0,0,1,0,0.09,255,255,1,0,0,0,0,0,0,normal,21
0,tcp,private,REJ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,121,19,0,0,1,1,0.16,0.06,0,255,19,0.07,0.07,0,0,0,0,1,1,neptune,21
0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,166,9,1,1,0,0,0.05,0.06,0,255,9,0.04,0.05,0,0,1,1,0,0,neptune,21
0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,117,16,1,1,0,0,0.14,0.06,0,255,15,0.06,0.07,0,0,1,1,0,0,neptune,21
0,tcp,remote_job,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,270,23,1,1,0,0,0.09,0.05,0,255,23,0.09,0.05,0,0,1,1,0,0,neptune,21
0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,133,8,1,1,0,0,0.06,0.06,0,255,13,0.05,0.06,0,0,1,1,0,0,neptune,21
0,tcp,private,REJ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,205,12,0,0,1,1,0.06,0.06,0,255,12,0.05,0.07,0,0,0,0,1,1,neptune,21
```

Рисунок. 3.1. Приклад даних з вхідного файлу

Згідно до рисунку 3.1, ми можемо побачити трансформацію запитів до єдиного формату, такий підхід дозволяє проводити аналіз даних та опрацьовувати їх.

```
uniquecolumns.remove('protocol_type')
uniquecolumns.remove('service')
uniquecolumns.remove('flag')
uniquecolumns.remove('someInt')
uniquecolumns.remove('attackType')
print(uniquecolumns)
feature_columns = []
for head in uniquecolumns:
  feature_columns.append(feature_column.numeric_column(head))

protocol_hot = feature_column.indicator_column(protocol_type)
service_hot = feature_column.embedding_column(service_type, dimension=70)
flag_hot = feature_column.indicator_column(flag_type)
#demo(service_hot)
feature_columns.append(protocol_hot)
feature_columns.append(service_hot)
feature_columns.append(flag_hot)
```

Рисунок 3.2. Створення заголовків, для визначення полів даних, що визначають клас об’єкту.

Згідно з малюнком 3.2. визначено значущі стовпці, для числових значень не потрібно створювати перечислення можливих значень, позаяк значення можуть знаходитись у межах від -нескінченності до нескінченності. А для категоріальних значень, тобто таких, що можуть набувати лише певних значень, необхідно спочатку визначити усі можливі варіанти, а потім, на основі цих варіантів дати визначення стовпцю.

```
uniquecolumns.remove('protocol_type')
uniquecolumns.remove('service')
uniquecolumns.remove('flag')
uniquecolumns.remove('someInt')
uniquecolumns.remove('attackType')
print(uniquecolumns)
feature_columns = []
for head in uniquecolumns:
  feature_columns.append(feature_column.numeric_column(head))
```

Рисунок 3.3. Опис роботи з датасетом

Приклади таких операцій з числовими стовпцями можна побачити на малюнку 3.3. Усі категоріальні стовпці та стовпці, що не визначають клас об’єкту видалено, потім усі стовпці, що залишились додаються до фінального списку стовпців.

```
#replacing unique values with matrix
protocol_type = feature_column.categorical_column_with_vocabulary_list('protocol_type', uniqueProtocolType)
service_type = feature_column.categorical_column_with_vocabulary_list('service', uniqueService)
flag_type = feature_column.categorical_column_with_vocabulary_list('flag', uniqueFlag)
```

Рисунок 3.4. визначення унікальних значень

```
protocol_hot = feature_column.indicator_column(protocol_type)
service_hot = feature_column.embedding_column(service_type, dimension=70)
flag_hot = feature_column.indicator_column(flag_type)
#demo(service_hot)
feature_columns.append(protocol_hot)
feature_columns.append(service_hot)
feature_columns.append(flag_hot)
```

Рисунок 3.5. Робота з категоріальними даними

Для категоріальних стовпців спочатку, на рисунку 3.4. визначаються унікальні значення кожного з стовпців, потім на рисунку 3.5. значення та усі можливі варіанти передаються до фінального списку стовпців.

Після цього, за допомогою усіх значень формується перший рівень нейронної мережі – рівень вхідних даних, у нашому випадку цей рівень складається з 41 елементу, серед яких 3 є категоріальними, тобто такі, що мають чітко визначені значення, усі інші є числовими значеннями.

```
batch_size = 64
train_ds = df_to_dataset(train, batch_size=batch_size)
val_ds = df_to_dataset(val, shuffle=False, batch_size=batch_size)
test_ds = df_to_dataset(test, shuffle=False, batch_size=batch_size)

model = tf.keras.Sequential([
    fearture_layer,
    #layers.Dense(1, activation='relu'),
    layers.Dense(10, activation='relu'),
    #layers.Dense(82, activation='relu'),
    #layers.Dense(128, activation='relu'),
    layers.Dense(1, activation='sigmoid')
    #layers.Dense(1, activation='linear', input_dim = 22)
])
```

Рисунок 3.6. опис створення нейронної мережі

Batch_size відповідає за кількість елементів, що одночасно будуть подані на нейронну мережу, збільшення цього параметру дозволить подавати більшу кількість елементів одночасно, що зменшує час на тренування. Але при значному збільшенні цієї змінної різко зменшується точність.

Задачею цієї роботи є створення додатку, здатного виявляти вторгнення. Через використання нейронних мереж неможливо точно сказати чи є запит небезпечним або безпечним, проте програма може зазначити свою точність та вірогідність кожного з рішень.

Як було зазначено у першому розділі, серед найбільш популярних атак з поміж 23 вирізняються 5, до того ж через величезну кількість класів атак, вигідніше змінити тип додатку з багатокласової класифікації на бінарну. Це дозволило підвищити точність з 63% до 96%.

```
1260/1260 [==============================] - 70s 55ms/step - loss: 0.4864 - accuracy: 0.9583 - val_loss: 0.5824 - val_accuracy: 0.9582
Epoch 2/3
1260/1260 [==============================] - 104s 83ms/step - loss: 0.7247 - accuracy: 0.9502 - val_loss: 0.7552 - val_accuracy: 0.9489
Epoch 3/3
1260/1260 [==============================] - 140s 111ms/step - loss: 0.4940 - accuracy: 0.9634 - val_loss: 0.3346 - val_accuracy: 0.9752
394/394 [==============================] - 33s 83ms/step - loss: 0.3138 - accuracy: 0.9769
```

Рисунок 3.7. точність моделі після тренування

3.2 Архітектура додатку

Структура моделі створена для передбачення значення вектору, який перетворений з запиту. Також структурою передбачена можливість додавання нових елементів до бази, що робить можливою модифікацію нейронної мережі та підвищення точності.

За сценарієм, користувач робить декілька запитів на сервер, система перетворює цей запит у вектор X, потім модель робить припущення Y, потім пара значень X та Y, пропонується адміністратору на аналіз. На малюнку 3.3 описана структура системи.

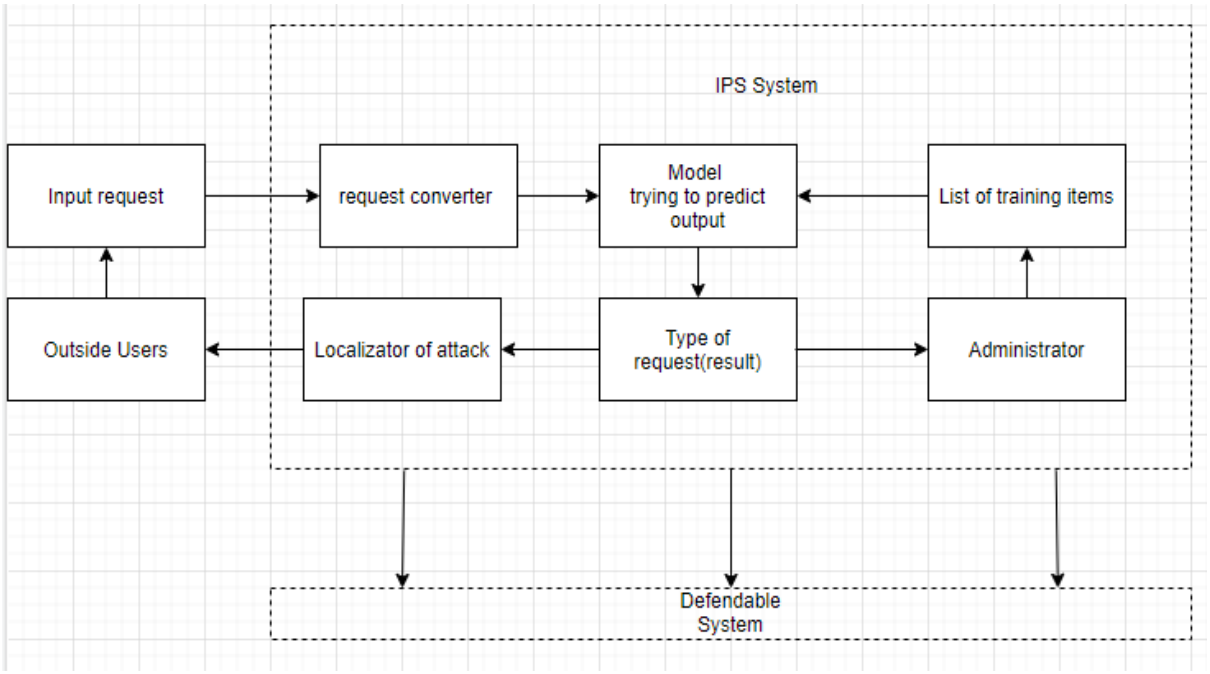


Рисунок 3.8. Архітектура програми

Таблиця 3.1. Опис кожного елементу архітектури

Елемент архітектури	Опис
Outside Users	Клієнт або сервіс, що робить запит та може бути заблокований системою
Input request	Група запитів, створених користувачем, якщо запит безпечний, він буде пересланий до системи клієнта
Request converter	Траснформує вхідні вектори у необхідний формат для моделі
Model	Отримує змінений вектор та список тренувальних даних, робить припущення, що може бути використано або адміністратором для підвищення точності моделі, або припущення може причиною для блокування користувача.
Елемент архітектури	Опис
Type of request	Головний показник для запиту, що одночасно з трансформованим запитом можуть бути використаними адміністратором для покращення моделі та самої системи. Також тип запиту визначає чи є запит небезпечним для сервісу.
Administrator	Може додавати нові елементи до списку тренувальних даних також аналізує вихідні дані системи(точність припущень) для того, щоб вручну знайти нові типи атак та підвищити точність системи.
Localization attack block	Блокує небезпечних користувачів та шукає аномальну поведінку користувачів

Далі опишемо алгоритм взаємодії елементів між собою. Необхідно передбачити можливість модифікувати тренувальні приклади. Також необхідно передбачити зберігання зроблених припущень, задля того, щоб у майбутньому можна було віднайти новітні небезпеки, про які на даний момент мало інформації.

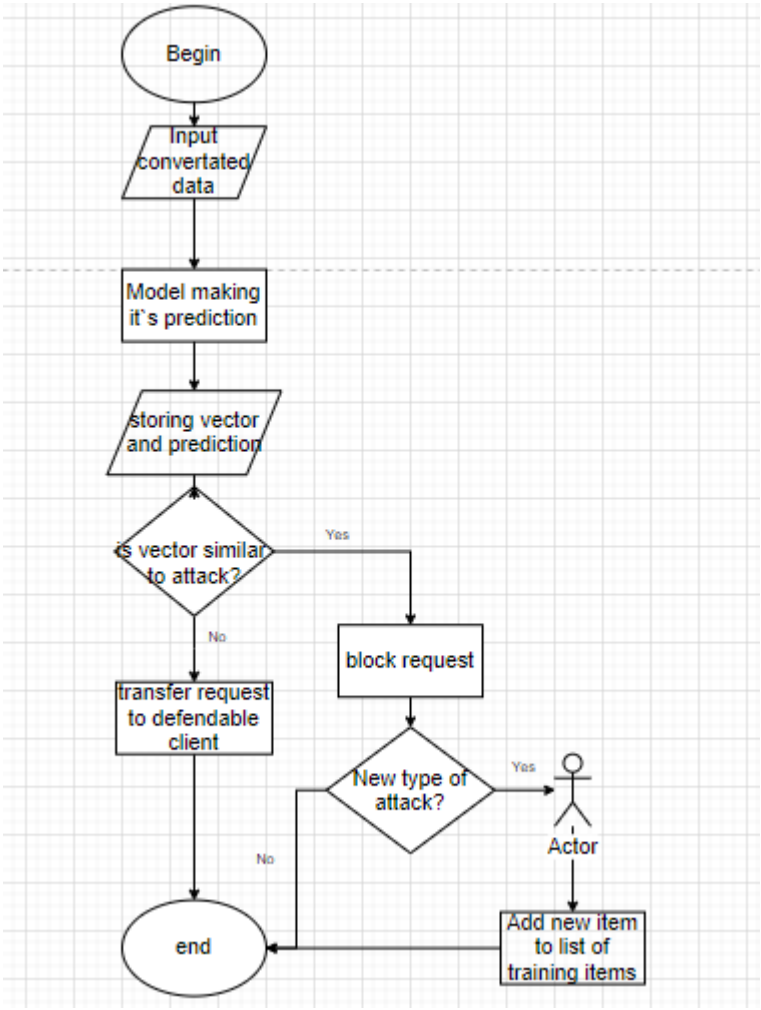


Рисунок 3.9. Алгоритм обробки запиту

Як можна побачити на малюнку 3.9, нові атаки відстежуються за допомогою адміністратора, тобто розробка повністю автономної системи є завданням на майбутнє.

3.3 Покращення моделі

Один з можливих засобів покращення точності може бути створення нових елементів або зменшення кількості найбільш кількісних елементів. Генерація нових елементів можлива за допомогою двох підходів:

- Генерація нових елементів на основі вже існуючих, наприклад, середнє або максимальне, або мінімальне значення кожного з полів певного класу.
- Отримання нових нових елементів під час роботи, які після аналізу адміністратором можуть доповнити існуючий тренувальний датасет.

Для реалізації першого методу можемо скористатись бібліотекою imblearn та функцією з неї SMOTE(Synthetic Minority Over sample Technique) [6], також можливий варіант ручної імплементації методів середніх максимальних та мінімальних значень.

Іншим направленням розвитку додатку, який є досить ефективним для систем з величезною кількістю користувачів, як наприклад соціальні мережі(facebook, instagram), сервіси для стрімінгу відео (twitch, youtubr, mixer) – створення безпекового профілю користувача, це дозволить виявляти аномалії у поведінці користувача, що може вказувати на те, що профіль користувача було викрадено. Однак автоматичне блокування таких користувачів призведе до різкого зменшення кількості акаунтів. Тому ефективним рішенням буде попередження адміністратора про вагомі зміни та тимчасово заблокувати користувача, сповістити його, та надати інструментарій для відновлення акаунту та посилення безпеки.

Для індикації необхідно встановити правила, що вказують на зміну звичного стану користувач:

- Зміна міста / країни входу, або зміна місця входу радіусом більше 40 км.
- Зміна звичного часу входу
- Зміна пристрою, з якого проведений вхід
- Зміна IP адреси
- Зміна частоти входу та кількості проведеного часу при користуванні сервісом.
- Зміна звичної поведінки користувача

Пошук аномалій у поведінці користувача дозволить виявляти не лише поведінкові зміни, що може вказувати на втрату акаунту користувача, але й на можливі модифікації пристрою, який використовує користувач.

Розглянемо інший підхід до машинного навчання – unsupervised learning. Для імплементації цього підходу використаємо алгоритм KNN(k nearest neighbors). Серед необхідних приготувань необхідно привести дані нормалізувати, для цього використаємо бібліотеку sklearn.preprocessing.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(x_train)

x_train = scaler.transform(x_train)
x_test = scaler.transform(x_test)
```

Рисунок 3.10. трансформація даних.

Після цього перетворення дані будуть знаходитись на відстані [-1..1], це дозволить, використовуючи евклідову відстань, підвищити точність.

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(x_train, y_train)
```

Рисунок 3.11. використання алгоритму KNN для навчання без вчителя

Тепер необхідно розглянути залежність точності від числа k.

Наприклад при k =5, отримаємо точність у 76 %

	precision	recall	f1-score	support
back	0.97	0.98	0.97	172
buffer_overflow	0.67	0.57	0.62	7
guess_passwd	0.93	1.00	0.96	13
imap	1.00	1.00	1.00	1
ipsweep	0.98	0.99	0.98	733
land	1.00	1.00	1.00	6
multihop	0.00	0.00	0.00	2
neptune	1.00	1.00	1.00	8282
nmap	0.98	0.96	0.97	272
normal	1.00	1.00	1.00	13400
perl	0.00	0.00	0.00	1
phf	0.00	0.00	0.00	2
pod	1.00	0.97	0.99	37
portsweep	0.99	0.99	0.99	599
rootkit	0.00	0.00	0.00	2
satan	0.99	0.98	0.98	749
smurf	1.00	1.00	1.00	556
teardrop	1.00	1.00	1.00	160
warezclient	0.93	0.96	0.94	197
warezmaster	1.00	1.00	1.00	4
accuracy			1.00	25195
macro avg	0.77	0.77	0.77	25195
weighted avg	1.00	1.00	1.00	25195

Рисунок 3.12. Приклад результату при k = 5

back	0.99	0.99	0.99	180
buffer_overflow	0.40	0.50	0.44	4
ftp_write	0.00	0.00	0.00	1
guess_passwd	1.00	0.88	0.93	8
imap	0.00	0.00	0.00	2
ipsweep	0.99	0.99	0.99	753
land	0.83	1.00	0.91	5
loadmodule	0.00	0.00	0.00	0
multihop	0.00	0.00	0.00	1
neptune	1.00	1.00	1.00	8172
nmap	0.97	0.97	0.97	285
normal	1.00	1.00	1.00	13543
perl	0.50	1.00	0.67	1
pod	0.97	1.00	0.99	35
portsweep	1.00	0.99	0.99	589
rootkit	0.00	0.00	0.00	3
satan	0.98	1.00	0.99	731
smurf	1.00	1.00	1.00	563
spy	0.00	0.00	0.00	2
teardrop	1.00	1.00	1.00	160
warezclient	0.92	0.96	0.94	153
warezmaster	1.00	0.75	0.86	4
accuracy			1.00	25195
macro avg	0.66	0.68	0.67	25195
weighted avg	1.00	1.00	1.00	25195

Рисунок 3.13. Приклад результату при k = 3

Згідно з рисунком 3.13, зменшення кількості сусідів, взятих до уваги, негативно впливає на точність класифікації та не зменшує час на необхідні операції.

	precision	recall	f1-score	support
back	0.96	0.99	0.97	190
buffer_overflow	0.33	0.50	0.40	2
guess_passwd	1.00	1.00	1.00	12
imap	0.00	0.00	0.00	1
ipsweep	0.98	0.99	0.98	727
land	0.57	1.00	0.73	4
loadmodule	0.00	0.00	0.00	4
neptune	1.00	1.00	1.00	8298
nmap	0.97	0.97	0.97	284
normal	1.00	1.00	1.00	13396
perl	0.00	0.00	0.00	1
phf	0.00	0.00	0.00	1
pod	1.00	1.00	1.00	50
portsweep	0.99	0.99	0.99	583
rootkit	0.00	0.00	0.00	2
satan	0.99	0.97	0.98	732
smurf	1.00	0.99	0.99	544
spy	0.00	0.00	0.00	1
teardrop	1.00	1.00	1.00	182
warezclient	0.91	0.97	0.94	178
warezmaster	1.00	0.67	0.80	3
accuracy			0.99	25195
macro avg	0.65	0.67	0.65	25195
weighted avg	0.99	0.99	0.99	25195

Рисунок 3.14. Приклад результату при k = 7

Як можна побачити з експериментів, збільшення k або його зменшення може привезти до зниження точності, тому визначення оптимального k, що в середньому дорівнює 5, є важливою задачею.

Необхідно порівняти ефективність використання навчання без вчителя та навчання з вчителем.

Таблиця 3.2. Недоліки та переваги кожного з варіантів машинного навчання

Питання	Навчання без вчителя	Навчання з вчителем
Необхідність тренування перед використанням	-	+
Можливість змінити структуру нейронної мережі	-	+
Використання даних без перетренування	+	-
Необхідність при внесенні змін перетренувати мережу	-	+
Простота у проектуванні системи	+	-
Складність під час вибору алгоритмів	-	+
Широкий вибір алгоритмів	-	+
Складність обчислень після тренувань	+	-

Як можна побачити з таблиці, використання навчання без вчителя легше реалізувати, аніж навчання з учителем. Проте навчання з учителем надає більшу точність у випадку бінарної класифікації, а навчання без вчителя надає можливість кластеризвувати дані та є більш ефективним у випадку багатокласової класифікації.

Розглянемо використання кластеризації у випадки двійкової класифікації, при k = 5.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	13539
1	0.99	1.00	1.00	11656
accuracy			1.00	25195
macro avg	1.00	1.00	1.00	25195
weighted avg	1.00	1.00	1.00	25195

Рисунок 3.15. Використання кластеризації для двійкової класифікації.

Як можна побачити з рисунка 3.15, кластеризація дозволяє досягти майже 100% точності, але у той самий час, кластеризація викликає величезні затрати на обчислення відстані до кожного елементу. Незважаючи на те, що нові елементи можуть легко додаватися, через складність алгоритму $O(n)$, додавання нових елементів сповільнює роботу системи. На цьому тлі використання нейронних мереж дозволяє швидко класифікувати нові об’єкти а додавання нових елементів можна визначити заздалегідь та підміняти щойно натреновану модель замість старої. Для вирішення проблеми ж з кластеризацією можна запропонувати обчислення середніх значень кожного з класів.

ВИСНОВКИ ДО РОЗДІЛУ 3

У третьому розділі наведено приклад архітектури системи IPS, наведено приклад вхідного файлу, наведено можливі шляхи розвитку продукту, нові модулі, що зможуть підвищити як ефективність сервісу загалом, так і ефективність модулю, відповідального за визначення атак.

Серед ефективних заходів – створення профілю користувача, заміна алгоритму машинного навчання, пошук правил, що вказують на порушення у звичній поведінці користувача, що у свою чергу може означати викрадення акаунту.

РОЗДІЛ 4. ЕКСПЕРЕМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

Структура нейронної мережі значною мірою впливає на точність припущень, тому основною задачею під час створення додатку з використанням машинного навчання є визначення оптимальної структури мережі.

Наприклад, 128 – 128 - 1 – базова схема, що забезпечила точність 96%.

```
1260/1260 [==============================] - 79s 63ms/step - loss: 0.6201 -
accuracy: 0.9532 - val_loss: 0.4803 - val_accuracy: 0.9616
Epoch 2/3
1260/1260 [==============================] - 122s 97ms/step - loss: 0.4979 -
accuracy: 0.9644 - val_loss: 0.4931 - val_accuracy: 0.9642
Epoch 3/3
1260/1260 [==============================] - 166s 132ms/step - loss: 0.4900 -
accuracy: 0.9657 - val_loss: 0.5445 - val_accuracy: 0.9615
394/394 [==============================] - 39s 99ms/step - loss: 0.5335 -
accuracy: 0.9622
Accuracy 0.9621750116348267
```

Рисунок 4.1. Ілюстрація процесу тренування структруи 128-128-1

З рисунку 4.1 можна побачити, що така структура надає майже 100% вірогідність правильності припущеня, розглянемо інші структури.

Наприклад, 41-128-128-1 – 97% точності

```
1260/1260 [==============================] - 80s 64ms/step - loss: 0.4588 - accuracy: 0.9572 - val_loss: 0.3983 - val_accuracy: 0.9675
Epoch 2/3
1260/1260 [==============================] - 124s 99ms/step - loss: 0.3869 - accuracy: 0.9690 - val_loss: 0.3141 - val_accuracy: 0.9768
Epoch 3/3
1260/1260 [==============================] - 169s 134ms/step - loss: 0.3762 - accuracy: 0.9722 - val_loss: 0.3139 - val_accuracy: 0.9771
394/394 [==============================] - 40s 101ms/step - loss: 0.3717 - accuracy: 0.9731
Accuracy 0.9730502367019653
```

Рисунок 4.2 структура 41-128-128-1

Розглянемо іншу конфігурацію, 41-128-1 – 97% точності, але більш швидкі обчислення.

```
1260/1260 [==============================] - 85s 67ms/step - loss: 0.5417 - accuracy: 0.9523 - val_loss: 0.4024 - val_accuracy: 0.9678
Epoch 2/3
1260/1260 [==============================] - 133s 106ms/step - loss: 0.3725 - accuracy: 0.9705 - val_loss: 0.3291 - val_accuracy: 0.9726
Epoch 3/3
1260/1260 [==============================] - 188s 149ms/step - loss: 0.3144 - accuracy: 0.9752 - val_loss: 0.3338 - val_accuracy: 0.9725
394/394 [==============================] - 47s 118ms/step - loss: 0.3527 - accuracy: 0.9708
Accuracy 0.9707878828048706
```

Рисунок 4.3 – 41-128-1

Тепер розглянемо структуру 41-82-1 – 97% точність.

```
1260/1260 [==============================] - 89s 71ms/step - loss: 0.5519 - accuracy: 0.9522 - val_loss: 0.4604 - val_accuracy: 0.9642
Epoch 2/3
1260/1260 [==============================] - 139s 110ms/step - loss: 0.3578 - accuracy: 0.9701 - val_loss: 0.3391 - val_accuracy: 0.9739
Epoch 3/3
1260/1260 [==============================] - 193s 153ms/step - loss: 0.3307 - accuracy: 0.9738 - val_loss: 0.3353 - val_accuracy: 0.9724
394/394 [==============================] - 50s 127ms/step - loss: 0.3011 - accuracy: 0.9746
Accuracy 0.9746378064155579
```

Рисунок 4.4. 41-82-1

1 – 41 – 82 – 1 – 91%

```
1260/1260 [==============================] - 90s 71ms/step - loss: 0.2928 - accuracy: 0.8981 - val_loss: 0.2513 - val_accuracy: 0.9145
Epoch 2/3
1260/1260 [==============================] - 137s 108ms/step - loss: 0.2478 - accuracy: 0.9159 - val_loss: 0.2499 - val_accuracy: 0.9153
Epoch 3/3
1260/1260 [==============================] - 181s 143ms/step - loss: 0.2461 - accuracy: 0.9166 - val_loss: 0.2487 - val_accuracy: 0.9155
394/394 [==============================] - 42s 107ms/step - loss: 0.2500 - accuracy: 0.9145
Accuracy 0.9144671559333801
```

Рисунок 4.5. 1-41-82-1

Як можна побачити з рисунку 4.5. така конфігурація знижує точність.

41-1 – 97%

```
1260/1260 [==============================] - 84s 67ms/step - loss: 0.3680 - accuracy: 0.9642 - val_loss: 0.3511 - val_accuracy: 0.9732
Epoch 2/3
1260/1260 [==============================] - 132s 105ms/step - loss: 0.3839 - accuracy: 0.9690 - val_loss: 0.3569 - val_accuracy: 0.9721
Epoch 3/3
1260/1260 [==============================] - 177s 141ms/step - loss: 0.3269 - accuracy: 0.9743 - val_loss: 0.3354 - val_accuracy: 0.9756
394/394 [==============================] - 42s 107ms/step - loss: 0.3230 - accuracy: 0.9760
Accuracy 0.9760270118713379
```

Рисунок 4.6. 41-1

Як можна побачити з прикладів, структура нейронної мережі впливає не тільки на точність, а й на час тренування та на складність операцій. Тобто, на рисунку 4.6 вдалося досягти точності у 97 відстоків, коли на рисунку 4.1 точність складає 96 відстоків, але модель має додатковий прихований шар.

4.1 Використання розробленої системи для забезпечення бещпеки мереж

Припустімо, що будь-яку мережу можна представити у вигляді графу, як наприклад:

- Кільцева топологія
- Зірка

Але застосування таких топологій значною мірою впливає на відмовостійкість та на швидкодію мережі. Тому розглянемо більш складні топології та графи для визначення більш відмовостійких систем.

Для початку візьмемо граф де Бруйна як приклад для створення топології.

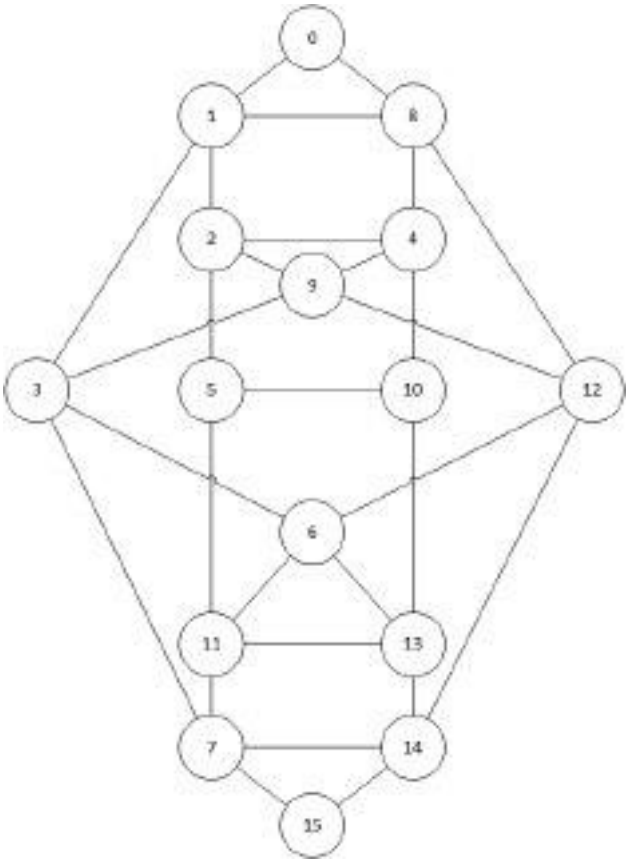


Рисунок 4.7. Граф де Бруйна

Припустимо, що усі дані входять до топології з точок 0, 3, 15 ,12. З цього виходить, що до будь-якої вершини топології можна дістатись максимум за 2 кроки.

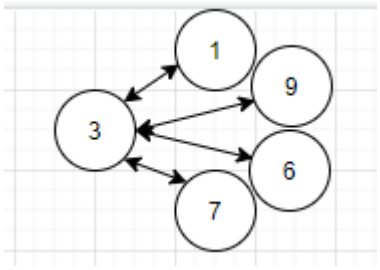


Рисунок 4.8. Ілюстрація взаємодії між елементами у топології

Для впровадження IPS для контролю за трафіком, що проходить скрізь мережу. Необхідно встановити модуль IPS в точку перетину максимальної кількості маршрутів даних.

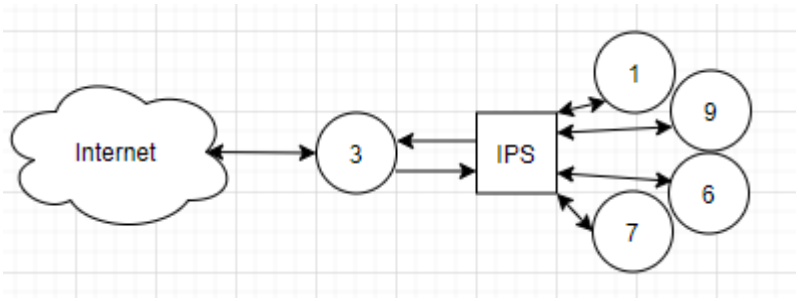


Рисунок 4.9. Структура топології після встановлення IPS

Такий варіант розташування IPS дозволяє контролювати увесь трафік у топології, заснованої на графі де Бруїна. Встановлення 4 таких модулів у точках 0, 3, 12, 15 дозволить залишити мобільність даних та трохи зменшить відмовостійкість системи.

Новою проблемою, що постає стало забезпечення взаємодії між елементами IPS, необхідно забезпечити єдину базу атак та можливість її оновлення кожною окремою систмою.

У наш час, модульні ситеми IPS та IDS використовують один модуль, як центральний, що відповідає за усі рішення, а усі інші – лише спостерігачі, що збирають інформацію. Я пропоную, кожному з модулів надати можливість діяти автономно від інших, а під час атаки, модуль буде сповіщувати інших та надіслатиме оновлену базу атак. Для синхронізації можна встановити часові інтервали, після яких модулі оновлюватимуть свою базу атак та обмінюватимуться новими даними.

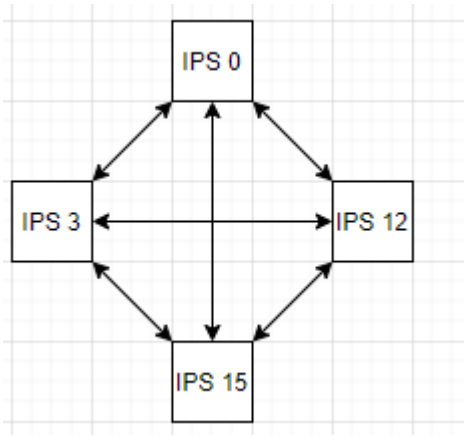


Рисунок 4.10. Взаємодія між елементами IPS

Така структура взаємодії між IPS зробить мережу більш відмовостійкою, та дозволить кожному окремому модулю, приклад якого зображено рисунку

4.9, працювати у повній ізоляції, у випадку, якщо усі інші частини мережі недоступні або скомпрометовані.Також такий модульний підхід дозволяє легко масштабувати системо, розглянемо на прикладі.

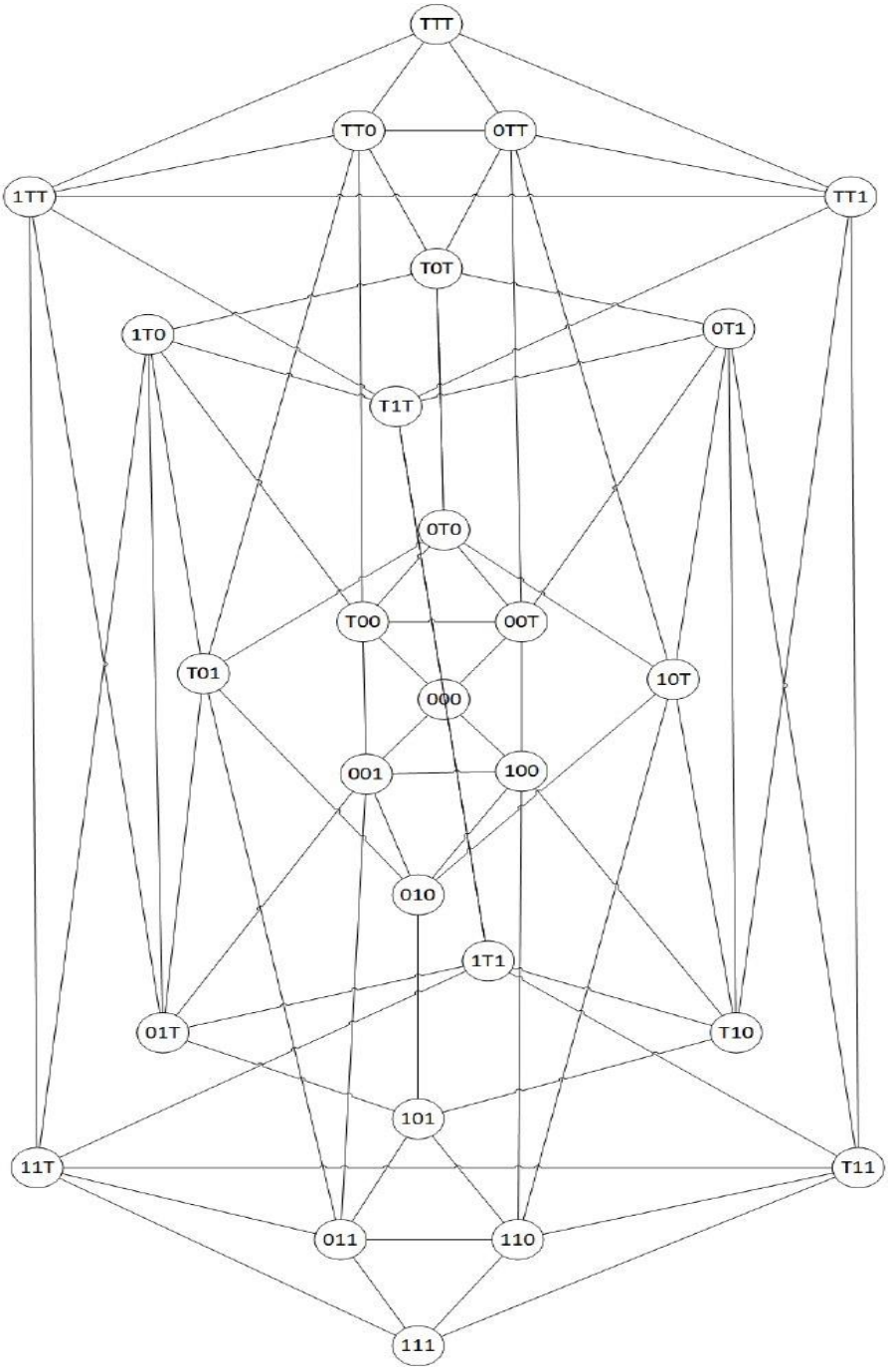


Рисунок 4.11. Масштабування системи.

Як можна побачити з рисунка 4.11, до будь якої вершини можна дістатись за три переходи, тому встановимо IPS блоки у вершинах 11T, T11, 1TT, TT1. Таким чином взаємодія між елементами такою самою, як на рисунку 4.10. Якщо забезпечити під’єднання до мережі тільки у точках, вказаних вище,

можна контролювати увесь трафік, що входить та виходить з мережі. Таким чином можливо виявляти не тільки загрози ззовні, а й шукати загрози усередині. Більша кількість модулів з IPS зроблять систему більш відмовостійкою, та здатною до відновлення, якщо вдасться ізолювати хоча б один елемент.

ВИСНОВКИ ДО РОЗДІЛУ 4

У розділі розглянуто експерименти зі зміни структури моделі, що показали цікаві результати, наприклад, зміна кількості шарів нейронної мережі майже не вплинула на її точність, це можливо є наслідком перетворення задачі з багатокласової класифікації на двійкову. Можливою причиною може бути overfitting, коли мережа натренувалася впізнавати приклади з тренувального датасету, але для уникнення такої ситуації, датасет був поділений на тренувальний та тестувальні набори.

ВИСНОВКИ

На прикладі задачі класифікації запитів були продемонстровані способи модифікації датасету для підвищення точності прогнозу.

Теоретично описано базис для створення нейронної мережі та наведені можливі недоліки при виборі алгоритму класифікації.

Наведені приклади засобів для розробки такого програмного продукту, показані результати кожного з підходів та порівняно їх ефективність, простоту та можливість модифікації для потреб кожного окремого користувача або розробника. Обґрунтовано вибір мови та засобів для програмування задачі.

Розглянуто можливі хмарні рішення як для розробки так і для публікації рішення. Описано вибір бази даних для збереження необхідної для функціонування проекту інформації.

Розглянуто архітектуру додатку, наведені приклади вже реалізованих систем, визначено найбільш важливі функції та модулі, що мають бути реалізовані, задля розробки IPS. Наведено переваги та недоліки використання трьох підходів для розробки додатку. Наведено процес розробки додатку та експерименти з покращення продуктивності та точності моделі.

Запропоновано до використання декілька підходів машинного навчання та після їх втілення наведено недоліки та переваги кожного з підходів та засоби вирішення недоліків або підвищення продуктивності.

У четвертому розділі розглянуто доцільність використання наведеної системи для запропонованих топологій, наведено можливі точки встановлення необхідних елементів, обґрунтовано архітектурний підхід по забезпеченню взаємодії між окремими модулями IPS та розглянуті недоліки та переваги, що з’являються при використанні такого підходу.

СПИСОК ПОСИЛАНЬ

1. Matthew K. Thoughtful Machine Learning with python. A test-driven approach. – O`Reilly 2017. – 216 с.
2. Clarence Chio, David Freeman Machine Learning and Security Protecting Systems with data and algorithms. – O`Reilly 2018. – 385
3. Security Architecture Vulnerabilities [Електронний ресурс] – Режим доступу до ресурсу: https://resources.infosecinstitute.com/category/certifications-training/cissp/domains/security-engineering/security-architecture-vulnerabilities (request date 11.05.2020).
4. Rowland, Craig H. "Intrusion detection system." U.S. Patent No. 6,405,318. 11 Jun. 2002.
5. Classify structured data [Електронний ресурс] – Режим доступу до ресурсу: https://www.tensorflow.org/tutorials/structured_data/feature_columns?hl=ru (request date 11.05.2020).
6. Description of library generating new items [Електронний ресурс] – Режим доступу до ресурсу: https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.over_sampling.SMOTE.html (request date 7.05.2020)
7. Files as source to neural network [Електронний ресурс] – Режим доступу до ресурсу: https://web.archive.org/web/20150205070216/http://nsl.cs.unb.ca/NSL-KDD (request date 6.04.2020)
8. Examples of IPSs [Електронний ресурс] – Режим доступу до ресурсу: https://www.comparitech.com/net-admin/ips-tools-software/(request 11.05.2020)
9. Rowland C. H. Intrusion detection system : пат. 6405318 США. – 2002.
10. Jackson G. M. Intrusion prevention system : пат. 7458094 США. – 2008.

11. SPLUNK architecture [Електронний ресурс] – Режим доступу до ресурсу: https://www.edureka.co/blog/splunk-architecture/
12. OSSEC architecture [Електронний ресурс] – Режим доступу до ресурсу: https://www.talentica.com/blogs/hids-implementation-using-ossec/
13. Fail2Ban architecture [Електронний ресурс] – Режим доступу до ресурсу: https://www.semanticscholar.org/paper/A-process-to-transfer-Fail2ban-data-to-an-adaptive-Ford-Mallery/9ebd70a6ddd5129419e7c7d8b0ac68e558525d54
14. Zeek architecture [Електронний ресурс] – Режим доступу до ресурсу: https://docs.zeek.org/en/current/cluster/
15. MS SQL Server architecture [Електронний ресурс] – Режим доступу до ресурсу: https://www.guru99.com/sql-server-architecture.html#10.

ДОДАТОК 1

Система виявлення вторгнень

Схема структурна

ІАЛЦ. 467200.003 Д1

Аркушів 1

Київ 2020

Зм	Арк.	№ докум.	Підпис	Дата	ІАЛЦ. 467200.003 Д1		
Розробив		Сміщенко Б.О.			Схема структурна	Літ	Маса	Масштаб
Перевірив		Волокита А.М.						
Н. контроль		Симоненко В.П.			Дипломна робота	Арк. 65	Аркушів. 1
Затверд.						КПІ ФІОТ	

ДОДАТОК 2

Система виявлення вторгнень

Схема функціональна

ІАЛЦ. 467200.004 Д2

Аркушів 1

Київ 2020

Зм	Арк.	№ докум.	Підпис	Дата
Розробив		Сміщенко Б.О.		
Перевірив		Волокита А.М.		
Н. контроль		Симоненко В.П.		
Затверд.				

ІАЛЦ. 467200.004 Д2

Схема Функціональна

Літ	Маса	Масштаб

Арк. 67 | Аркушів. 1

Дипломна робота

КПІ ФІОТ

ДОДАТОК 3

Система виявлення вторгнень

Схема принципова

ІАЛЦ. 467200.005 Д3

Аркушів 1

Київ 2020

					ІАЛЦ. 467200.005 ДЗ	
Зм	Арк.	№ докум.	Підпис	Дата	Схема Принципова	Літ / Маса / Масштаб
Розробив		Сміщенко Б.О.				Арк. 69 / Аркушів. 1
Перевірив		Волокита А.М.				
Н. контроль		Симоненко В.П.			Дипломна робота	КПІ ФІОТ
Затверд.						

ДОДАТОК Б. КОДОВА БАЗА

```
%tensorflow_version 2.x

import tensorflow as tf
!pip install -q sklearn


from google.colab import files
uploaded = files.upload()

%tensorflow_version 2.x

import tensorflow as tf
!pip install -q sklearn


from google.colab import files
uploaded2 = files.upload()
```

- Завантаження файлів, з якими працюватиме нейронна мережа

```
import pandas as pd
```

– Код для тренування нейронної мережі

```
import io
import tensorflow as tf
from tensorflow import feature_column
from tensorflow.keras import layers
from sklearn.model_selection import train_test_split

import numpy as np


dataframe2 = pd.read_csv(io.BytesIO(uploaded2["Field Names.csv"]))
dataframe2.pop('type')
dataframe2 = dataframe2.append({"column": 'attackType'}, ignore_index=True)
dataframe2 = dataframe2.append({"column": 'someInt'}, ignore_index=True)

uniquecolumns = []

uniqueAttackType = []
uniqueProtocolType = []
uniqueService = []
uniqueFlag = []
#uniqueAttack = []

for x in dataframe2.values:
  if x not in uniquecolumns:
    uniquecolumns.append(x[0])

print(uniquecolumns)
```

```
dataframe = pd.read_csv(io.BytesIO(uploaded["KDDTrain+.csv"]), names = uniq
uecolumns)

dataframe.pop('someInt')



def df_to_dataset(dataframe, shuffle=True, batch_size=32):
  dataframe = dataframe.copy()
  labels = dataframe.pop('attackType')
  ds = tf.data.Dataset.from_tensor_slices((dict(dataframe), labels))
  if shuffle:
    ds = ds.shuffle(buffer_size=len(dataframe))
  ds = ds.batch(batch_size)
  return ds



duration = feature_column.numeric_column("duration")
#demo(duration)
#getting unique values from columns
for x in dataframe['protocol_type']:
    if x not in uniqueProtocolType:
        uniqueProtocolType.append(x)

for x in dataframe['service']:
    if x not in uniqueService:
      uniqueService.append(x)


for x in dataframe['flag']:
    if x not in uniqueFlag:
      uniqueFlag.append(x)

for x in dataframe['attackType']:
    if x not in uniqueFlag:
      uniqueAttackType.append(x)
#dataframe['attackType'] = [uniqueAttackType.index(item) for item in datafr
ame['attackType']]
dataframe['attackType'] = [(0 if item == 'normal' else 1) for item in dataf
rame['attackType']]

train,  test = train_test_split(dataframe, test_size = 0.2)
train, val = train_test_split(train, test_size = 0.2)

print(len(train), 'train examples')
print(len(val), 'validation examples')
print(len(test), 'test examples')

#replacing unique values with matrix
protocol_type = feature_column.categorical_column_with_vocabulary_list('pro
tocol_type', uniqueProtocolType)
```

```
service_type = feature_column.categorical_column_with_vocabulary_list('serv
ice', uniqueService)
flag_type = feature_column.categorical_column_with_vocabulary_list('flag',
uniqueFlag)

uniquecolumns.remove('protocol_type')
uniquecolumns.remove('service')
uniquecolumns.remove('flag')
uniquecolumns.remove('someInt')
uniquecolumns.remove('attackType')
print(uniquecolumns)
feature_columns = []
for head in uniquecolumns:
  feature_columns.append(feature_column.numeric_column(head))

protocol_hot = feature_column.indicator_column(protocol_type)
service_hot = feature_column.embedding_column(service_type, dimension=70)
flag_hot = feature_column.indicator_column(flag_type)
#demo(service_hot)
feature_columns.append(protocol_hot)
feature_columns.append(service_hot)
feature_columns.append(flag_hot)

fearture_layer = tf.keras.layers.DenseFeatures(feature_columns)

batch_size = 64
train_ds = df_to_dataset(train, batch_size=batch_size)
val_ds = df_to_dataset(val, shuffle=False, batch_size=batch_size)
test_ds = df_to_dataset(test, shuffle=False, batch_size=batch_size)

model = tf.keras.Sequential([
    fearture_layer,
    layers.Dense(128, activation='relu'),
    layers.Dense(128, activation='relu'),
    layers.Dense(1, activation='sigmoid')
    #layers.Dense(1, activation='linear', input_dim = 22)
])
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'],
              run_eagerly=True)

print(train.dtypes)
model.fit(train_ds,
          validation_data=val_ds,
          epochs=3)

loss, accuracy = model.evaluate(test_ds)
print("Accuracy", accuracy)
```